VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zvýšení efektivity projektového řízení při vývoji software s využitím týmových nástrojů Team Foundation Server 2010 a interního informačního systému

Increase efficiency of project management in software development team using tools Team Foundation Server 2010 and the internal information system

2012 Lukáš Kozák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Lukáš Kozák**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Zvýšení efektivity projektového řízení při vývoji software s využitím týmových nástrojů Team Foundation Server 2010 a interního informačního systému
Increase Efficiency of Project Management in Software Development Team Using Tools Team Foundation Server 2010 and the Internal Information System

Zásady pro vypracování:

Cílem práce bude zdokumentování a následná implementace propojení mezi informačním systémem firmy CID International a nástrojem Team Foundation Server 2010. Během práce postupujte podle těchto bodů:
1. Seznámení se s technologií TFS2010 pro projektové řízení pomocí metodik „MSF for Agile Software Development" a "SCRUM".
2. Seznamení se se strukturou interního informačního systému (název interního systému - Market).
3. Analýza objektového modelu TFS2010 a objektového modelu informačního systému Market.
4. Vytvoření aplikace pro datovou synchronizaci mezi informačním systémem Market a TFS 2010.

Seznam doporučené odborné literatury:

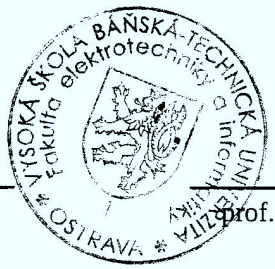
Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Lumír Návrat**

Datum zadání: 19.11.2010

Datum odevzdání: 04.05.2012



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 24. dubna 2012 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 24. dubna 2012 .....

Tímto bych chtěl poděkovat panu Petrovi Foltýnkovi z firmy CID International za zadání a praktické vedení práce a panu Ing. Lumírovi Návratovi za dohled nad jejím průběhem, formálním obsahem a cenné rady při tvorbě textu.

Abstrakt

Cílem práce je prostudovat možnosti podpory softwarového procesu v platformě TFS 2010 s použitím šablony MSF for Agile Software Development v5.0/Microsoft Visual Studio Scrum 1.0 a navrhnout a implementovat synchronizační nástroj mezi TFS 2010 a interním informačním systémem. Tím bude zvýšena efektivita projektového řízení ve firmě CID International, a.s., která využívá agilní metody pro vývoj software. Nezbytnou součástí práce je popis architektur a funkcí obou systémů.

Klíčová slova: Scrum, Team Foundation Server 2010, MSF for Agile Software Development v5.0, Microsoft Visual Studio Scrum 1.0, synchronizace, konflikt, časové razítko

Abstract

The goal of this thesis is to study options for software process support in the platform TFS 2010 using template MSF for Agile Software Development v5.0/Microsoft Visual Studio Scrum 1.0 and to design and implement a synchronization tool between TFS 2010 and an internal information system. That will increase effectivity in project management in the firm CID International, a.s. that uses agile methods for software development. Description of architectures and functions of both systems is necessary.

Keywords: Scrum, Team Foundation Server 2010, MSF for Agile Software Development v5.0, Microsoft Visual Studio Scrum 1.0, synchronization, conflict, timestamp

Seznam použitých zkratek a symbolů

CRM	–	Customer Relationship Management
HTTP	–	Hypertext Transfer Protocol
IDE	–	Integrated development environment
IIS	–	Interní informační systém
IoC	–	Inversion of Control
MS	–	Microsoft
MSF	–	Microsoft Solutions Framework
ORM	–	Object-Relational Mapping
SQL	–	Structured Query Language
TFS 2010	–	Team Foundation Server 2010
WIQL	–	Work Item Query Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Firma CID International používá Scrum jako agilní metodiku řízení projektů a má licencovaný Team Foundation Server 2010, který tuto metodiku přímo podporuje pomocí šablon MSF for Agile Software Development v5.0 a Microsoft Visual Studio Scrum 1.0. V první fázi tato práce uvádí čtenáře do problematiky agilního softwarového procesu do té míry, aby si vybudovali základ pro pochopení zbytku textu. Dalším bodem je srovnání zmíněných šablon a výběr vhodnější z nich. Nezbytná je i analýza objektových modelů obou systémů. Na základě výběru šablony, prostudování architektur obou systémů a seznámení se s obecnými možnostmi synchronizace bude možné navrhnout a implementovat synchronizační nástroj. Tato aplikace bude průběžně přenášet informace mezi nově používaným TFS 2010 a informačním systémem Market, který slouží k řízení a monitorování projektů nyní. To firmě CID International umožní začít plně využívat možností TFS 2010 v oblasti projektového řízení, protože nasazená aplikace umožní fungování obou systémů zároveň, a celkově tak bude zvýšena efektivita projektového řízení.

2 Agilní vývoj software

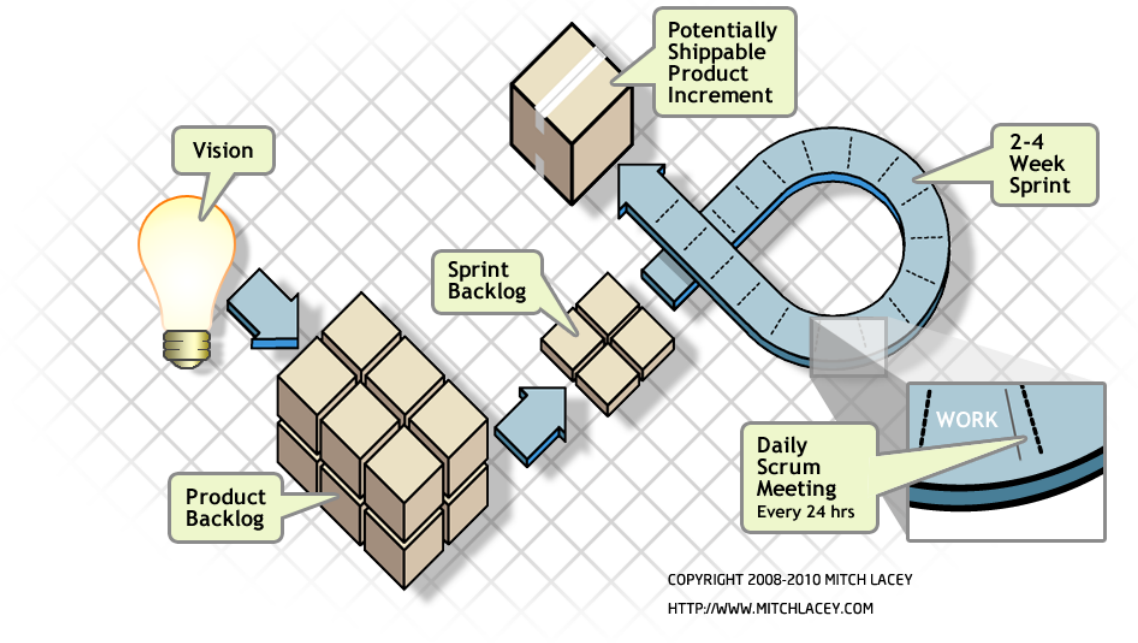
Agile development je pojem vyvozený z dokumentu zvaného *Manifest Agilního vývoje software* dostupného online [9]. Tento dokument byl sepsán skupinou lidí, která zahrnovala zakladatele Scrumu, Extrémního programování, Dynamic Systems Development Method, Crystalu a další myslitele z oblasti softwarového průmyslu. Tento manifest stanovil společný soubor základních hodnot a principů všech agilních metodik té doby:

- Jednotlivci a interakce před procesy a nástroji
- Fungující software před vyčerpávající dokumentací
- Spolupráce se zákazníkem před vyjednáváním o smlouvě
- Reagování na změny před dodržováním plánu

Těchto základních hodnot je dosahováno pomocí 12 principů:

- Naší nejvyšší prioritou je vyhovět zákazníkovi časným a průběžným dodáváním hodnotného softwaru.
- Vítáme změny v požadavcích, a to i v pozdějších fázích vývoje. Agilní procesy podporují změny vedoucí ke zvýšení konkurenceschopnosti zákazníka.
- Dodáváme fungující software v intervalech týdnů až měsíců, s preferencí kratší periody.
- Lidé z byznysu a vývoje musí spolupracovat denně po celou dobu projektu.
- Budujeme projekty kolem motivovaných jednotlivců. Vytváříme jim prostředí, podporujeme jejich potřeby a důvěřujeme, že odvedou dobrou práci.
- Nejúčinnějším a nejefektnějším způsobem sdělování informací vývojovému týmu z vnějšku i uvnitř něj je osobní konverzace.
- Hlavním měřítkem pokroku je fungující software.
- Agilní procesy podporují udržitelný rozvoj. Sponzoři, vývojáři i uživatelé by měli být schopni udržet stálé tempo trvale.
- Agilitu zvyšuje neustálá pozornost věnovaná technické výjimečnosti a dobrému designu.
- Jednoduchost–umění maximalizovat množství nevykonané práce–je klíčová.
- Nejlepší architektury, požadavky a návrhy vzejdou ze samo-organizujících se týmů.
- Tým se pravidelně zamýšlí nad tím, jak se stát efektivnějším, a následně koriguje a přizpůsobuje své chování a zvyklosti.

Tyto hodnoty a principy jsou funkční a byly ověřeny praxí. Každá z výše uvedených agilních metodik dosahuje těchto hodnot lehce jiným způsobem, ale všechny metodiky mají specifické procesy a praktiky, které ctí jednu nebo více hodnot.



Obrázek 1: Scrum Framework Flow Diagram

2.1 Scrum

Aby měl čtenář ponětí o obsahu zbytku práce, je nutné seznámit se s agilní metodikou vývoje softwaru zvanou Scrum. Informace o Scrumu jsou převzaty ze Scrum Alliance [13]. Scrum je metodika řízení projektů založená na principech a hodnotách z Manifestu agilního vývoje software. Ačkoliv byl původně utvořen pro softwarové projekty, funguje dobře i v jiných oblastech. Scrum je zdánlivě jednoduchý.

Na počátku projektu sestaví product owner seznam požadavků zvaný *Product Backlog*, který je seřazen podle priority. Během sprint planning meetingu si tým ze seznamu vybere některé položky (*Product Backlog Item*[1]) s největší prioritou a rozhodne se, jak je implementovat. Scrum tým má na dokončení těchto položek přesně stanovenou dobu, tzv. *Sprint*, obvykle 2 až 4 týdny. To je znázorněno na obrázku 1.

Tým se schází každý den, aby posoudil momentální stav sprintu a pokrok, kterého bylo dosaženo od poslední schůzky. Na průběh sprintu dohlíží scrum master, jehož úkolem je zajistit to, že se tým přibližuje k cílům, ke kterým se zavázal. Na konci sprintu je potenciálně doručitelná (*Potentially Shippable*) verze projektu, která je připravena na předání zákazníkovi nebo k uložení k pozdějšímu dokončení. Sprint je ukončen mítinky *Sprint Review* a *Sprint Retrospective*. Když začíná nový sprint, tým si zvolí další části ze seznamu a vše začíná znovu.

[1] v praxi nejčastěji ve formě User Story; tyto pojmy bývají zaměňovány

Koncept Scrumu a jeho terminologie jsou jednoduché, avšak je těžké je implementovat. Opravdového úspěchu Scrumu je dosaženo u takových týmů a organizací, které se ztotožňují s hodnotami a principy, které tvoří základ všech agilních procesů (viz Agile Manifesto, kapitola 2).

Následují 3 upřesnění entit ze Scrumu, o kterých jste se dočetli výše:

Product Backlog Je *DYNAMICKÝ*. Jednotlivé položky (*Product Backlog Item*) mohou být mazány a přidávány kdykoliv během projektu. Je prioritizovaný - nejdůležitější položky jsou implementovány nejdříve. Je postupně upřesňován - méně důležité položky nejsou záměrně rozpracovány do detailů.

Sprint backlog Jedná se o sadu položek z product backlgou, kterou se tým zaváže dokončit v průběhu momentálního sprintu. Tyto položky jsou rozporcovnány na konkrétní úkoly (*Task*) tak, aby je tým mohl splnit. Tým během sprintu spolupracuje na kompletaci položek ze sprint backlogu a schází se každý den (*Daily Scrum*), aby byly probrány překážky a pokrok, kterého bylo dosaženo. Na základně těchto denních schůzek je aktualizován sprint backlog a *burndown chart.*

Potentially Shippable Znamená, že část produktu je připravena k předání zákazníkovi. Rozhodnutí o tom, kdy nějakou část produktu předat, je na product ownerovi.

Ve scrumu se objevují tyto tři role:

Product Owner - zodpovědný za business hodnotu projektu

Scrum Master - zajišťuje produktivitu a správné fungování týmu

Scrum Team - samo-organizovaný tým developerů pracujících na projektu

Během sprintů se konají tyto druhy mítinků:

Sprint Planning - setkání týmu a product ownera, kde se vybírá podmnožina položek z product backlogu pro aktuální sprint

Daily Scrum - každodenní schůzka scrum týmu, kde se diskutuje o problémech a odvedené práci

Sprint Review - demonstrace odvedené práce product ownerovi po konci sprintu

Sprint Retrospective - ohlédnutí za uplynulým sprintem, kde tým navrhuje, jak zlepšit produkt i proces jeho vývoje

A konečně, tři artefakty Scrumu:

Product Backlog - prioritizovaný seznam výstupů a vlastností projektu

Sprint Backlog - podmnožina vybraných položek z product backlogu rozdělených na dílčí úkoly pro tým

Burndown Chart - graf, který nabízí rychlý pohled na to, kolik práce zbývá (může být doplněn burndown grafem pro celý projekt)

3 TFS 2010

3.1 Možnosti TFS 2010

Team Foundation Server 2010 od softwarové společnosti Microsoft je navržen pro týmový vývoj aplikací nad platformou .NET. Nabízí následující okruhy funkcí:

- Source control - správa zdrojových kódů
- Work tracking - řízení paralelního toku prací
- Reporting - zprávy o pokrytí kódu testy aj.
- Project management - projektové řízení
- Automated build process - automatizovaná tvorba buildů

Pro tuto práci jsou zajímavé okruhy Work tracking a Project management.

3.2 Architektura TFS

Architekturu TFS dělíme do těchto 3 logických celků:

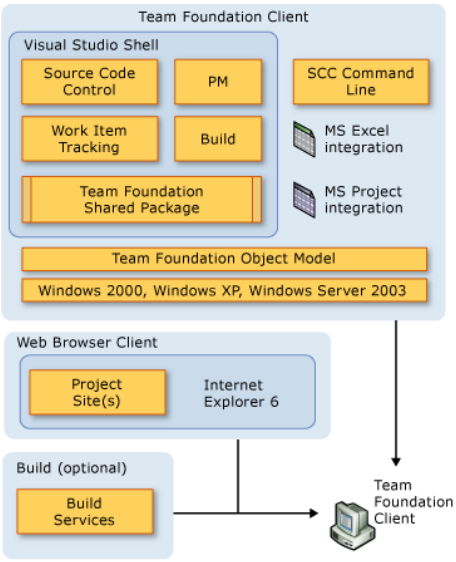
- Klientská vrstva
- Aplikační vrstva
- Datová vrstva

Klienti pracují s aplikační vrstvou přes různé webové služby. Příkladem klientů mohou být různé verze Visual Studia, MS Office spolu s pluginy, příkazová řádka... Funkce aplikační vrstvy jsou vystavovány přes HTTP protokol. Funkce můžeme rozdělit do dvou základních skupin - integrační funkce a datové funkce, přičemž pro nás budou podstatné ty datové. Obsahují 3 důležité okruhy funkčností:

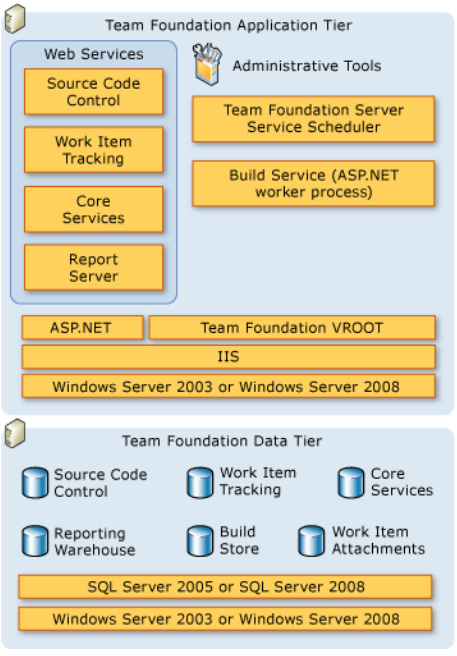
- Work item service
- Source code control service
- Build data services

Aplikační vrstva komunikuje s různými databázemi ve vrstvě datové. Datová vrstva TFS2010 nativně pracuje s MSSQL Serverem 2008, který je proto potřebný k chodu TFS2010. Při větším požadavku na výkon mohou být tyto vrstvy nainstalovány na více fyzických serverech, přičemž pro každý fyzický server je potřeba jedna licence.

Projekty mají nastaveny implicitní skupiny a oprávnění, které korespondují s běžnými rolemi ve vývojářských týmech. Pracovní úkoly jsou centrálně zaznamenávány, řízeny a udržovány v databázi pracovních úkolů. Tato centralizace ulehčuje všem členům týmu přístup k pracovním úkolům. Projektový manažer může ke správě databáze pracovních



Obrázek 2: Klientská logická vrstva TFS2010



Obrázek 3: Aplikační a datová logická vrstva TFS2010

úkolů používat Microsoft Office Excel a Microsoft Office Project. TFS 2010 poskytuje službu hlášení (reports), která zpracuje informace o stavu pracovních úkolů, výsledky buildů, výsledky testů a pokrytí testů a vygeneruje z nich komplexní přehled o stavu celého projektu. Pro každý projekt TFS 2010 vytvoří portál pro Microsoft Windows SharePoint®.

K projektovému řízení slouží v TFS 2010 šablony (process templates), jejichž dva exempláře budou následně analyzovány.

4 MSF for Agile Software Development v5.0

TFS 2010 obsahuje built-in šablonu označovanou jako MSF for Agile Software Development v5.0. Podle oficiálního popisu firmy Microsoft (zdroje [11], [12]) tato šablona implementuje praktiky a postupy popsané v dokumentu Agile Manifesto (viz kapitola 2). Měla by pomoci Scrum týmům aplikovat Scrum a praktiky z agilního vývoje software.

Projekt se v této šabloně skládá z *Iterations*, což odpovídá sprintu z terminoligie scrumu. Kromě přehledů jako *Active Bugs, Active Tasks, Open Issues* atd. se pod každou iterací nachází *Product backlog*, který je základním dokumentem každého projektu, potažmo sprintu.

4.1 Work Items

Work items zastřešují společné vlastnosti user stories, tasks, bugs... Slouží ke komunikaci a spolupráci v týmu. Je možné vytvářet z nich hierarchie a spojovat je, aby byly lépe stopovatelné. Šablona pro agilní metodiku obsahuje tyto druhy work items:

Bug - Závada nebo odchylka mezi očekávaným a skutečným chováním aplikace

Issue - Potenciální riziko, překážka nebo riziko v projektu

Shared Step - Skupina testovacích kroků, které mohou být sdíleny mezi jednotlivými Test cases

Task - Jednotka práce, která má být vykonána zaměstnancem nebo skupinou zaměstnanců

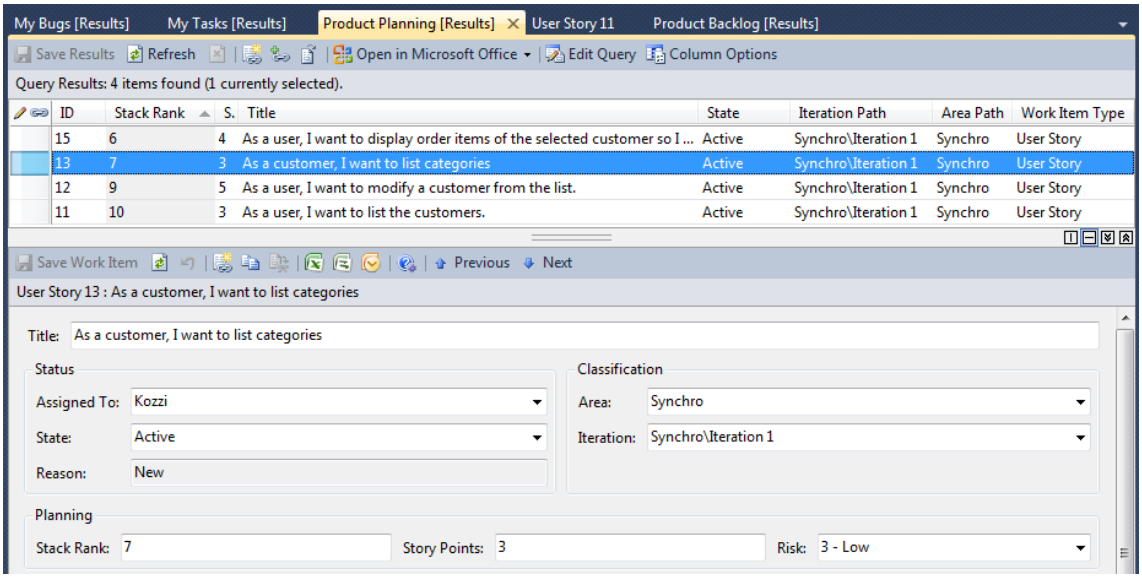
Test Case - Sekvence kroků včetně očekávaných výsledků; používá se ke zjištění správné funkčnosti projektu

User story - Popis hlavních rysů, funkcí a požadavků k implementaci vedoucích k uspokojení potřeb uživatele

Každý pracovní úkol má v databázi tyto atributy:

- ID
- Typ úkolu
- Stav úkolu
- Přiřazení k zaměstnanci

Pracovní úkoly nabývají několika stavů. User Story má tyto tři stavy: Active (dokud testy selhávají), Closed (testy procházejí) a Resolved (kód je kompletní a testy procházejí). Test Case má stavy Closed, Design, Ready. Task, Shared Steps a Issue mají pouze stavy Active a Closed, Bug má Active a Resolved.



Obrázek 4: Tvorba product backlogu ve VS2010

4.2 Team queries

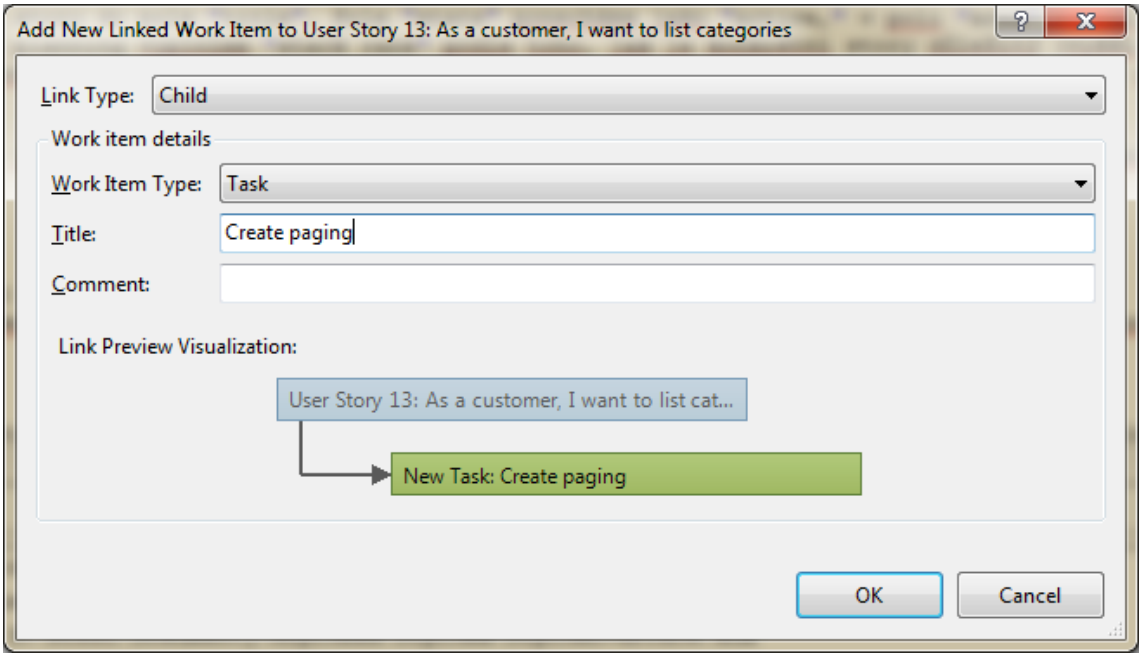
4.2.1 Product Backlog

Tvorba backlogu, který se skládá z user stories (tento pojem odpovídá pojmu Product Backlog Item ze Scrumu), je prvním krokem před každou iterací. Plánování je možné provést přímo ve Visual Studiu, přes Excel nebo SharePoint. Při plánování je možné využít hierarchie, aby backlog byl přehledný a bylo jasné, kam který úkol a story patří.

4.2.2 Iteration Backlog

Tento pojem odpovídá Sprint Backlogu z názvosloví Scrumu. Zobrazuje User Stories pro zvolenou Iteraci (a je tak podmnožinou Product Backlogu) včetně jednotlivých Tasks. Podívejme se na to, jak by v praxi probíhalo plánování sprintu. Na začátku vytvoříme novou iteraci nebo využijeme předem vygenerovanou iteraci a přejdeme k user stories. Při tvorbě jednotlivých stories vyplníme název user story do pole *title*. Pole *state* ponecháme jako *active*, v poli *area* by měl být název projektu. Nezapomeňme nastavit i iteraci, aby bylo jasné, do kterého sprintu story patří. V sekci planning vyplníme *stack rank* podle toho, jak je konkrétní story důležitý (vyšší číslo znamená větší důležitost). Důležitým prvkem je i pole *story points*, které reprezentuje časovou náročnost story. Pole *risk* vyjadřuje pravděpodobnost, že se časová náročnost story znatelně zvýší v průběhu iterace a tým bude muset reagovat způsobem, který nebyl v plánu. Ne nepodobným způsobem bychom vytvářeli úkoly a přiřazovali je ke stories.

Po rozdrobení stories na jednotlivé konkrétní úkoly je možné tyto úkoly (task) hierarchicky přiřadit ke stories. To provedeme kliknutím na příslušnou story a výběrem *New*



Obrázek 5: Tvorba work items ve VS2010

linked work item. Zvolíme kategorii task a vybereme úkol. Postupně, jak jsou v průběhu sprintu jednotlivé úkoly plněny, členové týmu akutalizují časové odhady pro úkoly (remaining time) a mění jejich statusy až do konce sprintu.

4.3 Reports

Burndown grafy v této šabloně nejsou vázány na iterace (které neexistují jako typ workitem), ale automaticky se jako počáteční a koncové datum zvolí období současného měsíce nebo období v okolí současnosti. Řešením této situace je změnit ručně všechny reporty tak, aby data odpovídala skutečné délce iterace. Po těchto změnách je možné docílit obdobných burndown grafů jako u druhé šablony (viz 5.3) kromě Velocity, který ovšem nepatří mezi klíčové reporty pro Scrum.

5 Microsoft Visual Studio Scrum 1.0

Předlohou této šabloně byla MSF for Agile Software Development v5.0. Pozorný čtenář už si jistě položil otázku typu: „Proč Microsoft další šablonu přímo pro Scrum, přestože původní šablona měla plnit stejný úkol?“ Podle Microsoftu ([3]) byla tato šablona vytvořena na základě podnětů těch klientů, kteří chtěli těsnější vazbu na Scrum a přesnou terminologii. Následující kapitoly ukazují, co je zde jinak.

5.1 Work Items

Scrum šablona obsahuje 4 typy work items, které byly přepracovány z původní šablony, a 3, které zůstaly v původním stavu (označeny hvězdičkou). I tady je možné je uspořádat do hierarchie protože se stále jedná o klasické work items, jen obohacené o některé prvky.

Impediment - Potenciální riziko, překážka nebo riziko mimo projektu, které může ohrozit sprint

Product Backlog Item - Obecný název pro objekt z backlogu; vhodné jako obdoba user story

Sprint - Uchovává informace o sprintu

Task - Jednotka práce, která má být vykonána zaměstnancem nebo skupinou zaměstnanců

***Bug** - Závada nebo odchylka mezi očekávaným a skutečným chováním aplikace

***Test Case** - Sekvence kroků včetně očekávaných výsledků; používá se ke zjištění správné funkčnosti projektu

***Shared Steps** - Skupina testovacích kroků, které mohou být sdíleny mezi prvky předchozí skupiny (Test case)

Všimněme si především Sprintu, tedy konkrétní iterace, který je tady nově implementován přímo jako work item. Má tyto důležité atributy:

Iteration - Uchovává název Sprintu. Pokud toto pole uložíme, bude vytvořen nový Sprint v odpovídající lokaci iteračního stromu, který je možné ve Visual Studiu 2010 zobrazit i editovat.

Start Date - Pole indikuje začátek Sprintu.

Finish Date - Pole indikuje konec Sprintu.

Sprint Goal - Popisuje hlavní cíl Sprintu, který by měl být měřítkem úspěchu při Sprint Review mítinku.

5.2 Team queries

5.2.1 All Sprints

Prvním dotazem je *All Sprints*, který zobrazí všechny sprinty v daném projektu. Zajímavým sloupcem je *Iteration Path*, který implicitně dělí projekt na releases a ty potom na jednotlivé sprinty. Cesta ke sprintům potom vypadá takto:

- Projekt\Release1\Sprint 1
- Projekt\Release1\Sprint 2
- ...

V závislosti na tom, jak scrum tým postupuje ohledně releasů, je možné toto chování změnit v Project Settings, Areas and Iterations, přičemž není nutné dodržet žádnou hierarchii. Jedinou podmínkou je, že cesta ke sprintu musí odpovídat jedné existující cestě z Areas and Iterations.

Důležitými atributy sprintu jsou *Start Date* a *End Date*, které vymezují sprint časově. Tato data potom slouží jako základ pro reporty, které jsou popsány v dalších kapitolách. V detailech sprintu je možné uvést *Sprint goal* a také přílohy v podobě souborů a komentářů. Posledním atributem je *Retrospective*, který obsahuje výstup sprint retrospective meetingu. Opět je možno přidat komentáře a přílohy.

5.2.2 Product Backlog

Pod tímto odkazem nalezneme všechny stories, které ještě nebyly implementovány a zařazeny do některého ze sprintů. Product owner by měl pro začátek specifikovat alespoň 2 atributy - prioritu (*Backlog priority*) a popis (*Description*). Dále by mělo být upřesněno pole *Effort*, které vyjadřuje prvotní časový odhad pro danou story. Při sprint planning meetingu tým doplní všechny chybějící údaje a přesune story do příslušného sprintu. Možnosti jak plánovat zůstávají stejné jako u první šablony.

5.2.3 Sprint Backlog

V této sekci najdeme všechny *stories, tasks*, případně *bugs* ze současného sprintu. To, který sprint momentálně probíhá, je vyvozeno z časových vymezení jednotlivých sprintů. Uživatelé Visual Studia 2010 se tedy nemusejí při běžné práci o výběr sprintu vůbec starat. Vše je zobrazeno hierarchicky v rozklikávacím seznamu (pokud tým postupoval správně při tvorbě backlogu). Mezi zobrazované atributy patří název, priorita, vykonavatel úkolu, stav, zbývající čas do dokončení, typ a informace, zda je úkol blokován. Obsah backlogu je možné řadit dle libovolného sloupce, například podle priority ...

Při sprint planning meetingu vývojový tým vytvoří jednotlivé úkoly a přidá časové odhady do pole *Remaining work* pro každý úkol. Je vhodné používat hodiny jako časovou jednotku, protože *Sprint burndown* graf, který ukazuje zbývající hodiny, bude později pracovat právě s tímto časovým údajem. Dále je vhodné přiřadit každému úkolu vztah *Child*

vzhledem ke story, ke které úkol patří tak, aby v backlogu vznikla přehledná hierarchie. Úkoly je možné rozdělit na podúkoly.

V průběhu sprintu si tým rozděluje úkoly z backlogu a při započetí úkolu se nastaví pole *Assigned to* na osobu, která úkol vykonává, a *State* se z *To Do* nastaví na *In Progress*. V ideálním případě vykonavatel s každým commitem snižuje hodnotu pole *Remaining work* dokud není práce hotova a nakonec nastaví *State* na *Done.* Samozřejmě se může stát, že úkol bude zablokován, případně úplně odstraněn, nebo bude *Remaining work* stagnovat nebo stoupat.

5.2.4 Blocked Tasks, Open Impediments

Blocked Tasks report obsahuje work items, které byly označeny jako *blocked,* ať už byl důvod jakýkoli. V případě, že se během sprintu objeví překážky, objeví se tady. *Open Impediments* zobrazuje přehled překážek, které momentálně blokují vývoj projektu. Je povinností scrum mastera starat se o tým tak, aby tento dotaz zobrazoval prázdný výsledek.

5.2.5 Test Cases, Unfinished Work, Work in Progress

Dotaz *Test Cases* zobrazí přehled všech test cases. Tento dotaz se nijak výrazně neliší od původní šablony. *Unfinished Work* ukazuje všechny *Product Backlog Items, Bugs a Tasks,* které ještě nebyly uvedeny do stavu *Done*, a s nimi spojené *Tasks.* Výsledkem je hierarchie stories a tasks, případně bugs, jako v product backlogu s tím, že work items ve stavu *Removed* nejsou brány v potaz. *Work in Progress* zobrazuje všechny úkoly z právě probíhajícího sprintu, které jsou ve stavu *In Progress.* Slouží jako přehled toho, co se na projektu právě dělá.

5.3 Reports

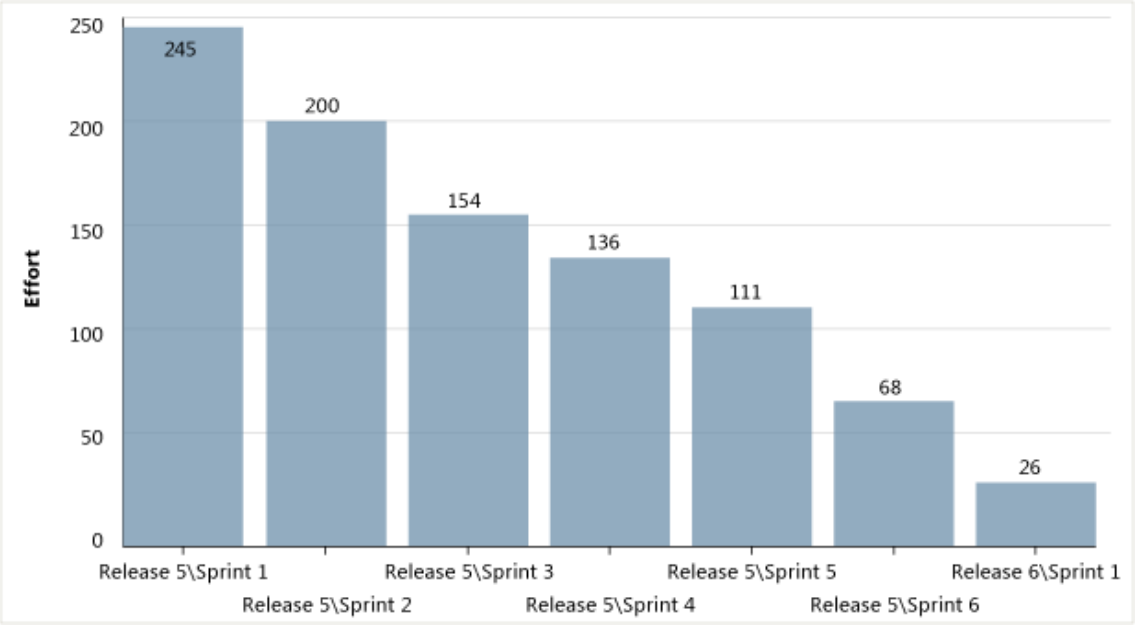
Stěžejními reporty pro scrum jsou *Release burndown, Sprint Burndown* a je zde i *Velocity,* který v Agile šabloně nenajdeme. Microsoft Visual Studio Scrum 1.0 nijak nesnaží o implementaci všech reportů z původní šablony. Scrum je jednoduše nepotřebuje.

5.3.1 Release burndown

Tento graf zobrazuje, kolik práce zbývalo na počátku každé iterace a jak rychle tým pracuje vzhledem k celkové práci z celého product backlogu. Zdrojem dat je product backlog. Na ose X vidíme jednotlivé sprinty a na ose Y součet veškeré práce (*Effort*) potřebného k vydání release. Data v grafu je možné filtrovat na základě sprintů a projektů.

5.3.2 Sprint burndown

Měřítkem úspěšnosti scrum týmu během sprintu je *Sprint Burndown* graf, který ukazuje, jak tým v průběhu času plní své závazky. Z tohoto grafu můžeme vyčíst, kolik práce



Obrázek 6: Ideální release burndown graf

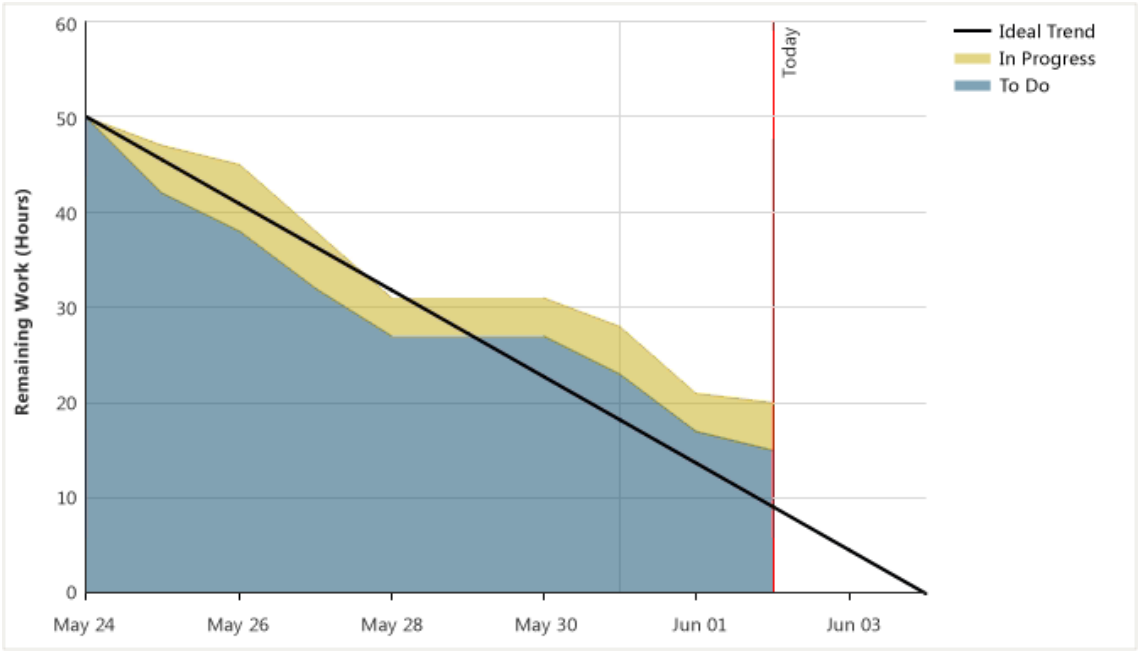
zbývá, jestli je tým schopen dodat všechny stories včas, kdy tým s úkoly skončí a kolik a celkové práce je momentálně rozpracováno. Podívejme se na obrázek 7.

Na ose Y vidíme zbývající práci, u které předpokládáme, že její množství bude s ubývat s časem (v hodinách) na ose X, který jí věnujeme. Toto množství je v grafu barevně odlišeno na úkoly, které ještě nebyly započaty (modrá), a úkoly, na kterých se momentálně pracuje (žlutá). Černá linka představuje ideální trend, podle kterého by zývající práce ubývala lineárně s ubíhajícem časem na ose X a tato osa by se protnula s ideální linkou přesně v moment, kdy je vše hotovo a čas vypršel. Červená úsečka *Today* ukazuje momentální den. Z grafu můžeme vyčíst, že scrum tým má v tomto okamžiku mnohem více zbývající práce, než by podle ideálu měl mít, a nestihne práci dokončit včas.

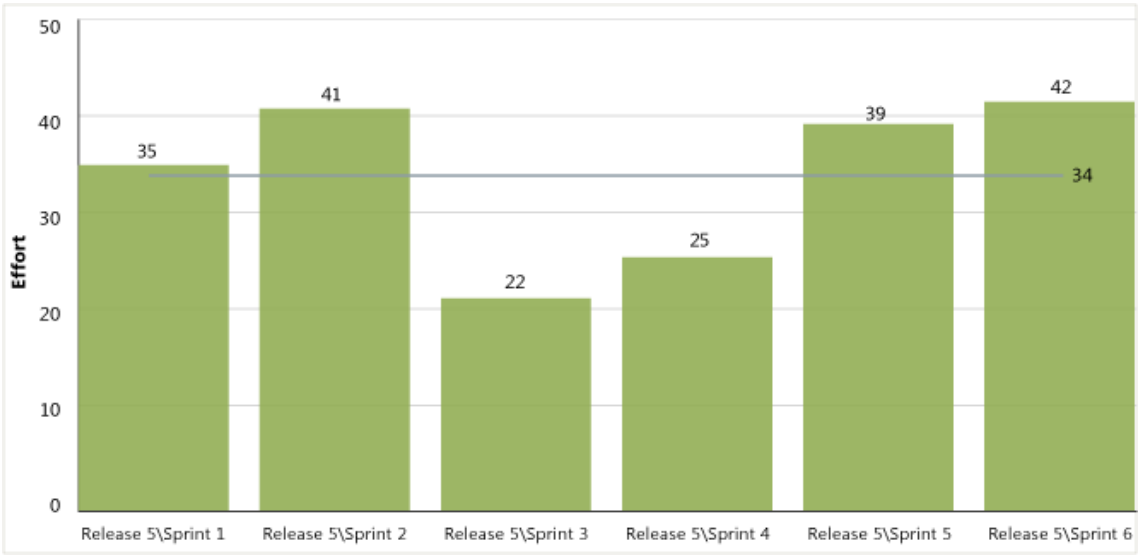
Důležité je vědět, že pokud se tým rozhodne rozdělovat úkoly na menší, časově méně náročnější podúkoly, bude muset specifikovat hodiny pouze pro podúkoly, aby nedošlo k započtení stejné práce dvakrát. Aby vše fungovalo a graf měl vypovídací hodnotu, musí jednotliví členové poctivě aktualizovat pole *Remaining work* a označovat úkoly jako ukončené v momentě, kdy je hotovo. To je však dobře vidět v backlogu, který je denně aktualizován v průběhu scrum daily meetingů.

5.3.3 Velocity

Posledním grafem je *Velocity*. Ukazuje, kolik práce (*Effort*) tým dokončil v jednotlivých sprintech. Z toho vidíme i minima, maxima a průměr (šedá horizontální linka). Zdrojem dat jsou product backlog items (stories) a bugs z product backlogu. Příkladem může být obrázek 8.



Obrázek 7: Příklad sprint burndown grafu



Obrázek 8: Příklad velocity grafu

6 Výběr šablony

Na začátek je nutné říct, že obě šablony je možné pro Scrum použít. MSF for Agile Software Development v5.0 je robustní, zatímco Microsoft Visual Studio Scrum 1.0 se zaměřuje na aspekty, které Scrum opravdu potřebuje. Rozdíl bude v pohodlí a časové náročnosti na jejich používání a na to musí být kladen důraz, protože cílem této práce je zefektivnit projektové řízení.

Vizuálně evidentní je rozdíl v terminologii, přičemž šabolna šitá na míru Scrumu ji dodržuje přesně, kdežto Agile šablona užívá obecných pojmů z agilního softwarového procesu. Některá pole navíc mohou být matoucí. Scrumu stačí u úkolů jeden časový údaj, a to je zbývající čas - z tohoto hlediska má Task v Agile šabloně nadbytek atributů (original, remaining, completed).

Fatálním rozdílem mezi šablonami je to, že Agile šablona naprosto postrádá časový koncept iterace - v reálném světě jsou Sprinty časově vyměřeny daty Od a Do. Z toho vyplývá, že Agile šablona neumožňuje rychlý a jednoduchý pohled na Sprint jako předem definovaný časový interval bez toho, aniž by před každým Sprintem byly změněny dotazy reportů tak, aby si data odpovídala a ukazovala správné grafy pro daný Sprint. Toto v případě kratších iterací efektivitě projektového řízení rozhodě nepřidá, nemluvě o nutnosti měnit data na všech místech v případě, že se termín změní. Tímto způsobem není informace o Sprintu vůbec uložena v TFS 2010. Proto Microsoft Visual Studio Scrum 1.0 zavádí nový druh work item - Sprint - který časové vymezení obsahuje a je jednoznačně uložen v TFS 2010. Na základě toho je možné bez námahy sledovat průběh Sprintů, protože burndown reporty s těmito daty pracují.

Vezmeme-li jednoduchost a striktnost Scrumu proti schopnostem obou šablon, zjistíme, že Microsoft Visual Studio Scrum 1.0 je to, co opravdu potřebujeme, a MSF for Agile Software Development v5.0 by mohlo být tím, co potřebujeme, kdybychom měli času nazbyt. Tím je zodpovězena otázka, proč Microsoft vydal druhou šablonu přímo pro Scrum a jakou šablonu vybereme pro projektové řízení a zbytek této práce - bude to Scrum šablona, protože chceme být efektivní.

Na závěr je dobré si říci, že existující projekty, vytvořené pod jinou šablonou, je možné importovat s pomocí nástroje *TFS Integration Platform.*

7 Market

7.1 Funkce Marketu

Market je CRM[1] systém, jehož součástí jsou moduly pro projektové řízení. Databáze obsahuje více než stovku tabulek, přičemž pro synchronizační nástroj je podstatných jen několik z nich.

7.2 Databáze Marketu

Databázovým serverem pro Market je MSSQL. Samotná báze dat obsahuje následující entity:

Zadání - O vzniku Zadání rozhoduje vedoucí realizačního týmu VRT (ve spolupráci s manažerem pobočky MP nebo vedoucím výroby VV). Zadání by zásadně mělo vzniknout nejpozději ve chvíli, kdy se manažer pobočky nebo vedoucí realizačního týmu dozví o uzavření závazné smlouvy (zpravidla bude vznikat dříve - už při obchodním jednání nebo při předběžných pracích). K jedné zakázce může vzniknout libovolný počet Zadání - podle provozní potřeby. Zadání může vzniknout buď tak, že ho pořídí k tomu oprávněný uživatel IIS, nebo může vzniknout prostřednictvím Helpdesku (v tomto případě ho musí následně zodpovědná osoba začlenit do systému tím, že mu přiřadí konkrétní zakázku). Jako databázová entita obsahuje časovou náročnost a popis práce, která je požadována. Je na ni vázána entita Požadavek ve vztahu 1:N, tj. jedno zadání má více požadavků.

Požadavek Obsahuje popis konkrétních úkolů pro řešitele.

Požadavek-výkon rozpis Tato entita slouží k rozdělení úkolů řešitelům a k jejich časovému naplánování, včetně odhadů potřebného času k vyřešení těchto úkolů. Je to tedy „Plán řešení."

Požadavek-výkon Zde se uchovávají informace o skutečné době strávené na provádění úkolů. Jedná se vlastně o „Skutečný postup řešení."

Číselníky Pro synchronizační nástroj jsou důležité i některé číselníky v databázi. Jsou to tyto tabulky: číselník stavu zadání, číselník typu práce, číselník stavu požadavku, číselník priority, číselník kategorie zadání.

[1] CRM - Customer Relationship Management; proces shromažďování, zpracování a využití informací o zákaznících firmy

8 Obecné možnosti synchronizace

Jak je patrné už z abstraktu této práce, cílem je i implementace synchronizačního nástroje. Proto je před designovou fází této aplikace vhodné prozkoumat teoretická řešení jiných, podobných aplikací, a v případě, že se ukáží jako vhodná, je aplikovat při vlastním programování. Na taková řešení se podíváme v této kapitole.

8.1 Kompletní a částečná synchronizace

Kompletní, neboli pomalá synchronizace spočívá v tom, že data jsou zkopírována ze serveru na klienta v momentě, kdy se klient synchronizuje poprvé, nebo když uplynul synchronizační interval. Ve druhém případě klient obsahuje neaktuální data, dokud neproběhne nová synchronizace.

Částečná, neboli rychlá synchronizace přenáší pouze podmnožinu všech dat, a to pouze ta data, která byla změněna. K uchování informací o těchto změnách se používá logging nebo change tracking. Částečná synchronizace může být provedena pouze v případě, že klient někdy v minulosti provedl plnou synchronizaci se serverem. Tyto informace je možné najít v publikaci [1].

8.2 Pesimistická a optimistická replikace

Tradiční techniky replikace se snaží udržovat jednu konzistetní kopii a dávají uživatelům iluzi, že pracují s identickou replikou. Tyto techniky ale blokují uživatele v případě, že replika není aktualizovaná. Z tohoto důvodu se tomuto způsobu synchronizace říká pesimistická. Například algoritmy pro primární kopie fungují tak, že si z replik zvolí jednu primární, která je potom zodpovědná za všechny přístupy k potřebnému objektu. Po aktualizaci primární replika zapíše změny synchronně do všech replik. Pokud primární replika selže, sekundární repliky si zvolí novou primární. Tyto techniky fungují dobře v LAN, kde je výborná latence a pravděpodobnost selhání minimální.

Druhou skupinou algoritmů jsou optimistické algoritmy. Hlavním rozdílem je přístup k uživateli v případě, že probíhá update. Zatímco pesimistické techniky uživatele zablokují, optimistické algoritmy umožní přístup k datům s tím, že pravděpodobnost konfliktů je velmi malá. Pokud se konflikt přesto objeví, je opraven posléze. Tento přístup umožňuje lepší dostupnost dat, je vhodnější pro implementaci v Internetu, umožňuje vznik více replik a jejich uživatelé zůstávají autonomní. Dále je umožněna asynchronní spolupráce mezi klienty (např. CVS) a lepší zpětná vazba, protože aktualizace se projeví hned po uložení. Tyto informace je možné najít v publikaci [2].

9 Požadavky

Firma se rozhodla, že primární nástroj pro plánování kapacity a přidělování úkolů bude TFS 2010 a Market bude sloužit jako reporting pro management. Nicméně synchronizační nástroj má fungovat oboustranně, tj. jestliže se něco změní v TFS, musí to být promítnuto do Marketu a naopak. Pokud se od poslední synchronizace změní položka jak v TFS, tak v Marketu, jedná se o konflikt, který musí být řádně zalogován tak, aby to uživatelé vzali na vědomí a problém po synchronizaci opravili. Řešením konfliktu v době synchronizace je nahrazení konfliktních dat v TFS 2010 daty z Marketu, tedy Market „vždy vyhrává." Plánování sprintů a tedy i tvorba celého backlogu ale proběhne v TFS 2010, takže při prvním spuštění synchronizačního nástroje musí být kompletní backlog zkopírován do Marketu. V případě smazání Product Backlog Item nebo Task z TFS 2010 synchronizační nástroj takovou situaci detekuje a zaloguje tak, aby uživatelé Marketu mohli danou položku smazat i v Marketu.

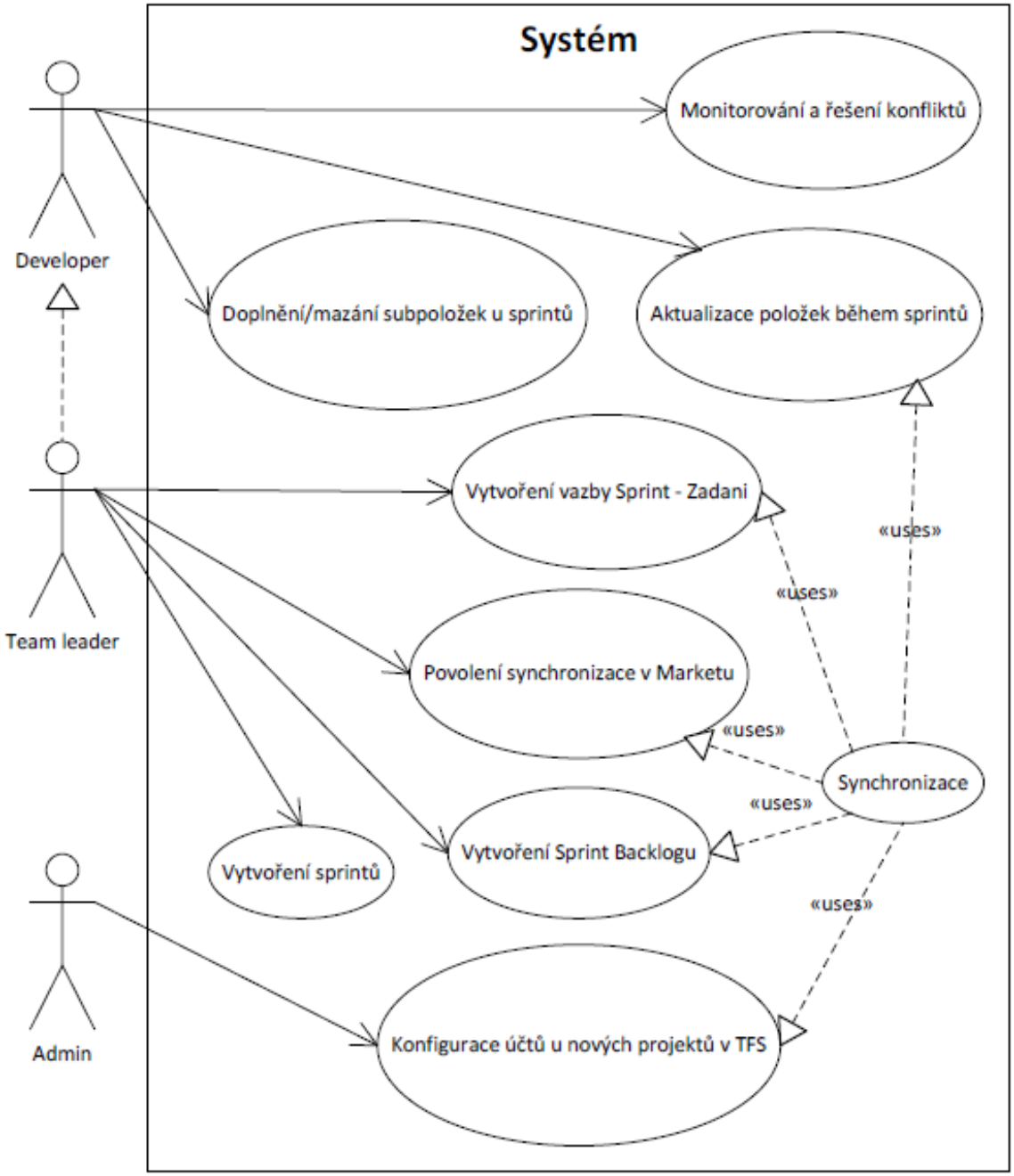
V konfiguraci nástroje musí být umožněno, aby byly synchronizovány pouze právě probíhající sprinty nebo všechny sprinty najednou. Není nutné často aktualizovat ukončené sprinty, jelikož data v nich se mění zřídkakdy. Důvodem je výkonnost aplikace, která by s příbývajícími daty nezanedbatelně klesala. Stejně tak nebudeme aktualizovat řádky, které se nezměnily.

Před začátkem nového sprintu vytvoří uživatelé nový Sprint v TFS 2010 pod příslušným projektem a záznam v tabulce Zadani a zkopírují jeho vygenerované ID do nového sprintu v TFS, čímž se vytvoří vazba Sprint-Zadani. Synchronizační nástroj umí vytvořit nové záznamy v tabulkách Pozadavek a PozadavekVykonRozp. Dále v tabulce Zadani existuje sloupec, který umožní vypnutí synchronizace pro celý sprint včetně product backlog items a tasks.

Vztáhneme-li daná fakta na kapitolu 8.1, jedná se o částečnou synchronizaci, přičemž za první plnou synchronizaci můžeme považovat první běh programu po naplánování sprintu, kdy proběhne prosté zkopírování entit z TFS 2010 do Marketu. [1] Ačkoliv se nejedná o replikaci, z hlediska optimistických a pesimistických přístupů (kapitola 8.2) se na vznikající synchronizační nástroj můžeme dívat jako na optimistický algoritmus, protože počet současně pracujících uživatelů bude minimální a pravděpodobnost konfliktů malá.

S ohledem na zvýšení efektivity projektového řízení bude nástroj pracovat samostatně s co nejmenším počtem vstupů od uživatelů. Synchronizační nástroj bude pracovat s assemblies od firmy Microsoft určenými pro práci s TFS 2010. Pro demonstrativní účely bude nástroj vytvořen jako konzolová aplikace s tím, že veškerá funkcionalita bude uložena v knihovnách tak, aby bylo možné snadno vytvořit jiný druh aplikace, jako například Windows Service. Interakci mezi aktéry z firmy a navrženým synchronizačním nástrojem ukazuje use case na obrázku 9.

[1] Nicméně, aplikace musí být schopna vytvořit nové položky kdykoliv.



Obrázek 9: Interakce mezi firmou a systémem

10 Analýza

Smyslem této kapitoly je zamyslet se nad netriviálními problémy spojenými s výrobou synchronizačního nástroje, které bude nutné vyřešit v implementační fázi.

10.1 Mapování entit Marketu na entity v TFS2010

Jak už je trošku patrné z předchozího textu, oba systémy mají mnoho společných funkcí. Oba dva dokáží evidovat formulace zadání od zákazníků, které se následně dělí na úkoly. Dále umějí plánovat práci s ohledem na prioritu úkolů, uchovávat informaci o času stráveném na konkrétním úkolu, pamatovat si stavy zadání i stavy jednotlivých kroků vedoucích k jejich splnění a další informace. K realizaci synchronizace těchto společně uchovávaných informací je potřeba identifikovat takové atributy v TFS 2010 a Marketu, které nesou společnou informaci. K tomu poslouží dokumentace obou systémů, případně konzultace s firmou.

10.2 Časová razítka

V TFS 2010 má každá work item (tedy i Sprint, Product Backlog Item a Task) vlastní historii změn, která je aktualizována při jakékoli operaci update provedené na dané work item. Každý záznam o změně obsahuje jméno pole, které bylo změněno, starou a novou hodnotu pole, číslo verze pole, autora změny a časové razítko (time stamp).

Naopak Market je navržen jednoduše, bez jakéhokoliv verzování nebo evidence časových razítek. Jakákoli změna je prostě uložena do příslušné tabulky a předchozí hodnota je nenávratně ztracena.

Máme-li implementovat obousměrnou částečnou synchronizaci, je potřeba si uvědomit, že k určení směru synchronizace (TFS do Marketu nebo naopak) je potřeba znát časové razítko poslední změny. Pomocí této informace můžeme určit, který systém obsahuje poslední platný údaj a aktualizovat příslušný záznam ve druhém systému. Jak už víme, Market žádné změny v čase nezaznamenává.

10.3 Konflikt

Za konflikt považujeme jev, kdy uživatelé změní záznam v obou systémech a není možné automaticky rozhodnout, který za záznamů je platný. U známých version control systémů konflikty řeší uživatelé, kteří jsou vyzváni k tomu, aby zvolili správný údaj. Synchronizační nástroj, který je předmětem implementační fáze této práce, něco takového nemá v požadavcích. Zadávající firma si za řešení konfliktů zvolila jednoduché přepsání informací v TFS 2010 informacemi z interního informačního systému. Market je tedy „dominantním“ systémem.

Podívejme se na to, proč samotný systém časových razítek (v případě Marketu) nestačí k detekci konfliktů. Řekněme, že poslední synchrnonizace náhodného např. úkolu proběhla v 8:00 (což je nyní uloženo v časovém razítku daného úkolu) a oba systémy jsou v rámci tohoto úkolu vzájemně konzistetní. V 8:10 uloží uživatel A změny do TFS 2010 a v 8:15

uloží uživatel B změny do Marketu, přičemž oba provedli UPDATE na stejném záznamu. Časové razítko v Marketu nyní ukazuje 8:15, čímž je ztracena informace o poslední synchronizaci v 8:00. Synchronizační nástroj je spuštěn v 8:30 a zjišťuje, že poslední platný UPDATE proběhl v 8:15 v Marketu a zvolí synronizaci směrem z Marketu do TFS 2010. Detekce konfliktu není možná, protože je ztracena informace o předposlední změně v Marketu (8:00) a UPDATE uživatele A v 8:10 není k čemu vztáhnout.

Triviálním řešením této situace by bylo druhé časové razítko ve sloupcích Marketu, které by uchovávalo informaci o předposlední změně. Tak bychom mohli porovnat provedené změny mezi 8:00 a 8:15, protože čas 8:00 by byl uchován v „předposledním razítku“, a zjistili bychom konflikt, který byl vyvolán operací UPDATE v TFS 2010 v 8:10. Představme si však situaci, kdy dojde navíc k další změně v Marketu v 8:20. Tato změna by nastavila časová razítka na 8:15 a 8:20 a konflikt z 8:10 by byl zapomenut. Řešením této situace by bylo třetí razítko, ale představme si, že by k další změně Marketu na stejném řádku došlo v 8:25. Potřebovali bychom 4. razítko…

Přidávání více a více časových razítek evidentně k řešení nevede. Přidáme-li však sloupec, který zaznamená **čas poslední synchronizace**, daný problém to vyřeší, protože nepotřebujeme podrobnosti o tom, jaké změny proběhly nebo kolik jich bylo. Stačí nám upozornění, že mezi poslední a současnou synchronizací došlo ke konfliktu. Vztáhneme-li toto na výše uvedený příklad, vždy bude k dispozici informace o tom, že poslední synchronizace proběhla v 8:00 a z toho je možné detekovat konflikt mezi časy 8:10 a 8:15, protože jsou v intervalu mezi současností (8:30) a poslední synchronizací. Jak se později ukázalo, k tomuto postupu došli také autoři reportu číslo [1], kteří jej popisují v už abstraktu svého díla. TFS 2010 obsahuje kompletní historii změn daného úkolu.

11 Architektura

Ještě před tím, než začneme aplikaci programovat, je nezbytné ze zamyslet nad její strukturou. Tato kapitola se zabývá návrhovými vzory, které jsou v aplikaci použité, a nakonec je prezentován výsledný třídní diagram.

11.1 Dependency Injection

Tímto tématem se podrobně zabývá Martin Fowler ve svém článku [6]. Tento a další odstavec jsou jen jeho nezbytným výtahem. Jedním z často používaných návrhových vzorů v objektově orientovaném programování je Inversion of Control. Ten se vyznačuje především de-couplingem, což znamená, že kód, který vykonává určitý úkol, je striktně oddělen od kódu, který řídí datové toky v aplikaci. Jednoduše řečeno - každý se soustředí na to, co má dělat. To byl hlavní důvod pro výběr právě tohoto vzoru, a je tím zaručena znovupoužitelnost kódu. Jednotlivé moduly neví nic o konkrétní implementaci jiných modulů a je tak vyloučeno, že by se vzájemně ovlivňovaly.

Technika Dependency Injection je jednou z možností, jak implementovat IoC. V sychnronizačním nástroji je implementována pomocí constructor injection, což znamená, že (téměř) každá třída očekává v parametrech konstruktoru(ů) rozhraní, nikoliv konkrétní třídy. DI tedy zahrnuje třídu, její závislosti popsané rozhraními a tzv. container, který na požádání vytvoří instanci třídy, která implementuje požadované rozhraní. Do konstruktoru tedy můžeme přes rozhraní vložit jakoukoli třídu bez nutnosti měnit kód. To je možné udělat i v době run-time, a je tak možné provést injekci libovolného množství různých tříd implementujících jedno rozhraní, a například tak otestovat různé implementace stejné softwarové komponenty.

K realizaci výše popsaného je v synchronizačním nástroji použit Microsoft Unity 2.1. Spárování tříd a rozhraní (injection) není v této aplikaci rozhodováno v době běhu programu, ale v konfigurační třídě nazvané Bootstrapper. Ve výpisu 1 je řečeno, že implementací rozhraní ITFSRepository bude třída TFSRepository. V dalším bloku je potom rozhraní ITFSRepository, jehož implementjucí třída už je zaregistrována v kontejneru, vloženo do konstruktoru další konkrétní třídy, a je tak vytvořena závislost. Tímto způsobem je nakonfigurována celá aplikace, která je jednoduše testovatelná a modulární.

```
container.RegisterType<ITFSRepository, TFSRepository>(
  new InjectionConstructor(
  new ResolvedParameter<WorkItemStore>()
));

container.RegisterType<IDeletedWorkItemDetector, DeletedWorkItemDetector>(
  new InjectionConstructor(
  new ResolvedParameter<IMarketUnitOfWork>(),
  new ResolvedParameter<ITFSRepository>()
));
```

Výpis 1: Ukázka IoC

11.2 Perzistentní vrstva - TFS

Tato vrstva je implementována pomocí návrhového vzoru Repository (viz článek [7]). V něm je k přístupu do TFS 2010 využito knihovny *Microsoft.TeamFoundation*, který poskytuje vše potřebné k realizaci synchronizačního nástroje. Informace, jak tuto knihovnu využít, je vhodné začít čerpat z odkazu [10].

Ke čtení dat z TFS 2010 je nutné znát dotazovací jazyk WIQL, který vychází z jazyka SQL. Repozitář pro přístup do TFS 2010 proto používá metody, které kombinují WIQL a C#, jako ve výpisu číslo 2.

```
public IEnumerable<IProductBacklogItem> GetProductBacklogItems(string iterationPath)
        {
            string wiqlQuery = string.Format(@"SELECT [System.Id], [System.WorkItemType], [
                System.Title], [System.AssignedTo], [System.State] " +
                @"FROM WorkItems WHERE [System.WorkItemType] = 'Product Backlog Item'
                    AND  [System.IterationPath] UNDER '{0}' ORDER BY [System.Id]",
                iterationPath
                );

            WorkItemCollection witCollection = workItemStore.Query(wiqlQuery);

            List<ProductBacklogItem> result = new List<ProductBacklogItem>();

            foreach (WorkItem item in witCollection)
            {
                var workItem = new ProductBacklogItem(item);
                result.Add(workItem);
            }

            return result;
        }
```

Výpis 2: Vrať všechny Product Backlog Items podle dané cesty ke Sprintu

Přístup k properties všech work items přes zmíněný jmenný prostor probíhá přes indexery následovně:

```
object a = workItem["Microsoft.VSTS.Scheduling.RemainingWork"];
```

Výpis 3: Způsob práce s atributy work items

Tento způsob poskytuje užitečnou flexibilitu v případě, kdy se do šablony pro TFS 2010 přidá nějaký nový atribut:

```
object a = workItem["Market.PozadavekVykonRozp.IDPozadavekVykonRozp"];
```

Výpis 4: Způsob práce s atributy work items

Programátor nemusí přidání nového atributu do TFS 2010 nijak zvlášť ošetřovat - stačí znát *Ref name* a datový typ tak, jak byly zadány do šablony. Vše je vyhodnoceno až za běhu programu. Představme si ale, že pokaždé, když v kódu potřebujeme přistoupit na nějaký atribut Sprintu, Product Backlog Item nebo Task, budeme muset použít zmíněný

indexer. Může snadno dojít k nechtěným překlepům, jelikož to, na jaký atribut přistupujeme, není v době kompilace známo. Dále, protože indexer vrací *object*, bylo by nutné pokaždé přetypovat:

```
int idPozadavekVykonRozp;

try
{
    Int32.TryParse(workItem["Market.PozadavekVykonRozp.
        IDPozadavekVykonRozp"].ToString(), out idPozadavekVykonRozp);
    return idPozadavekVykonRozp;
}
catch (Exception e)
{
    return null;
}
```

Výpis 5: Přetypování požadovaného atributu

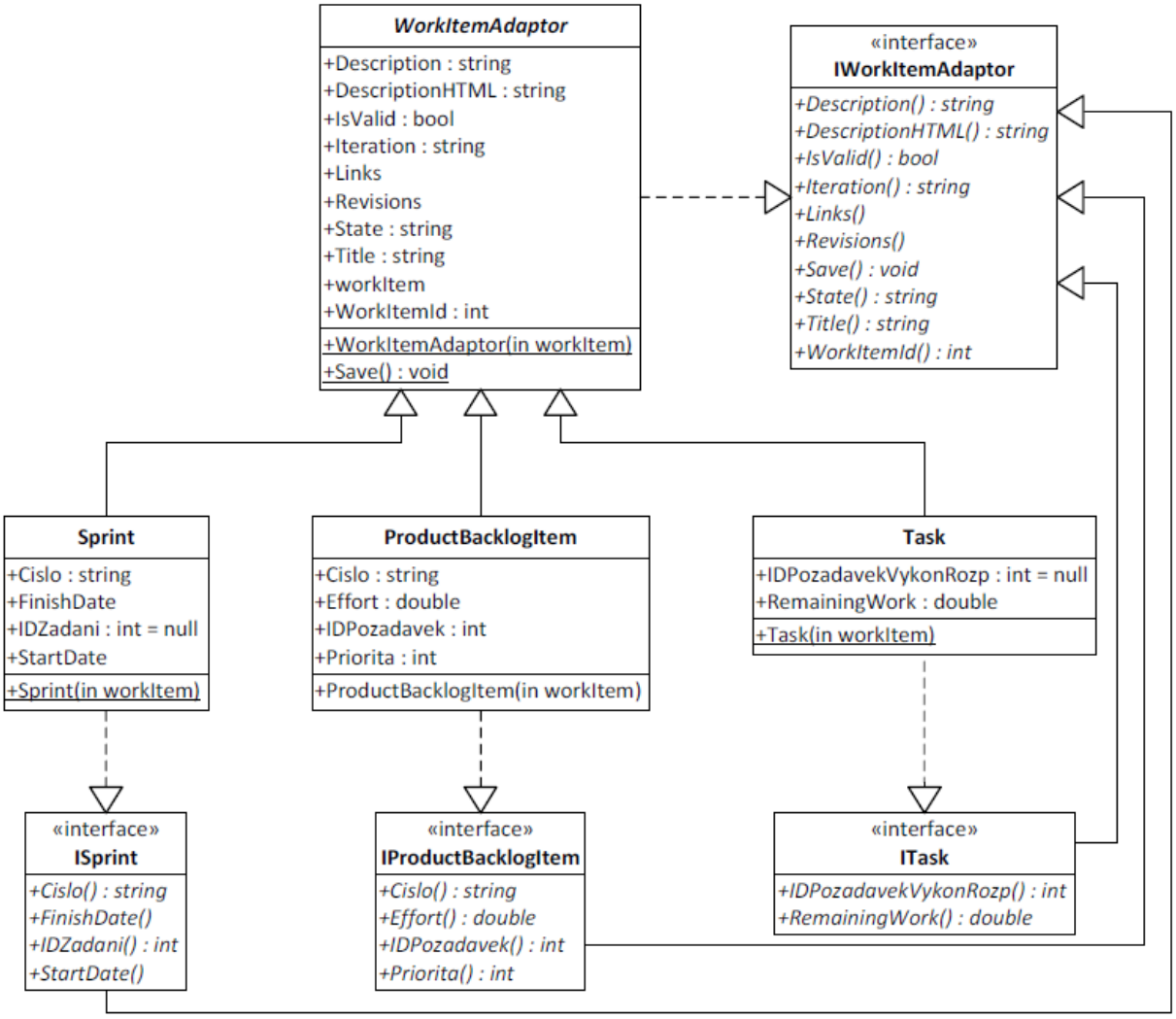
Takový přístup by vyústil v nechtěnou nepřehlednost kódu a těžkou údržbu aplikace. Proto je vhodné využít návrhový vzor Adapter (neboli Wrapper), který tento nepříjemný způsob práce s daty schová a poskytne dobře použitelné rozhraní. K tomuto účelu byly vytvořeny třídy Sprint, Product Backlog Item a Task a rozhraní popisující jejich chování. Je to patrné z obrázku 10. Všechny tři entity jsou v podstatě WorkItem a ty mají mnoho atributů společných. Z tohoto důvodu bylo přidáno rozhraní IWorkItemAdaptor, které tyto společné atributy zastřešuje, a třída WorkItemAdaptor, která k nim realizuje přístup (opět Adapter). Dědičností u ISprint, IProductBacklogItem a ITask z IWorkItemAdaptor je zajištěno, že třídy realizující Sprint, Product Backlog Item a Task budou poskytovat společné atributy. Třída WorkItemAdaptor je abstraktní, ale implementuje společné metody z IWorkItemAdaptor.

11.3 Perzistentní vrstva - Market

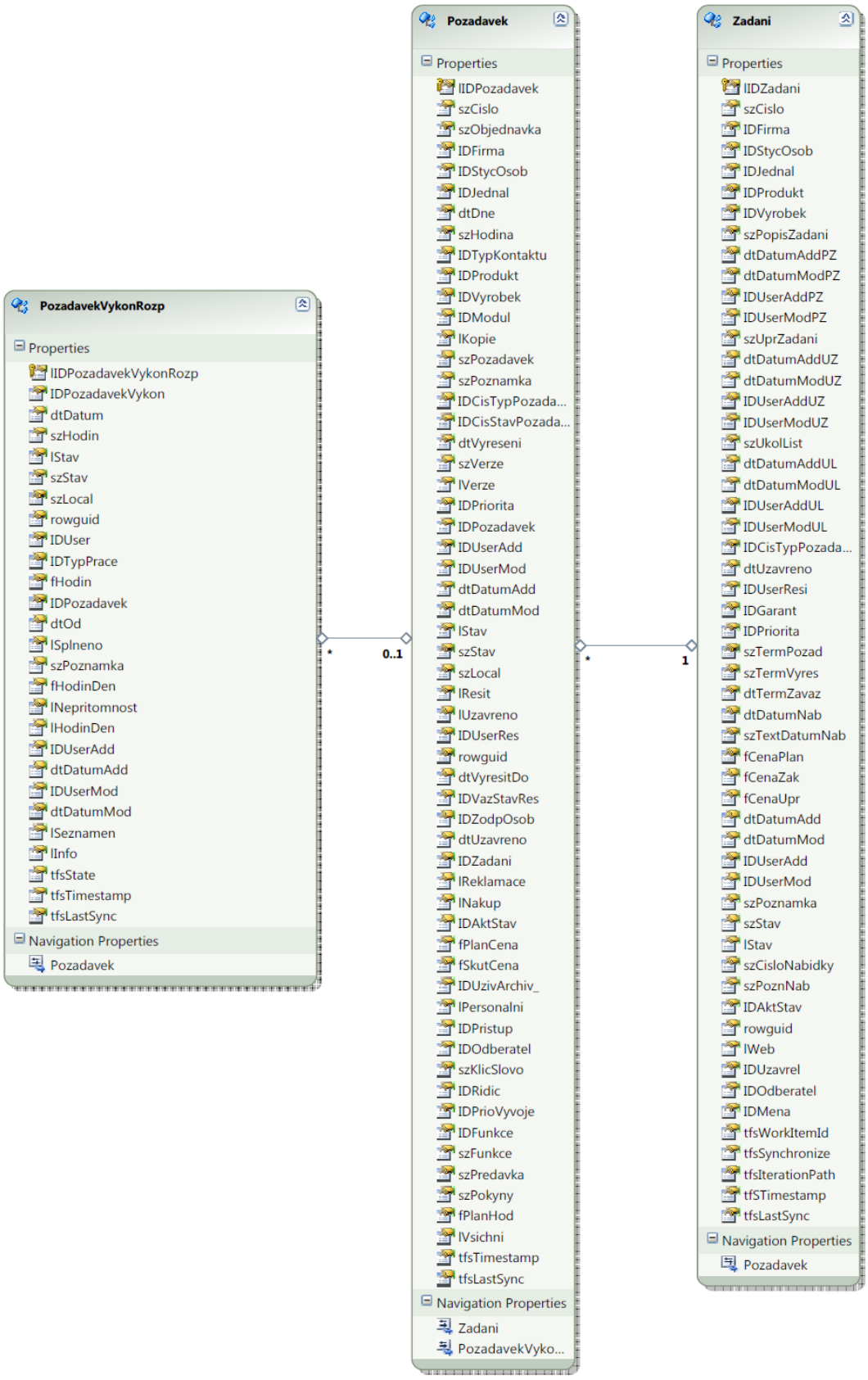
Výměna dat mezi Marketem a synchronizačním nástrojem je zajištena pomocí ORM frameworku ADO.NET Entity Framework, který umožňuje manipulovat s databázovými entitami jako s objekty. Tento objektový model (obrázek 11) je používán nepřímo přes rozhraní IMarketUnitOfWork implementující třídu MarketUnitOfWork. Viz obrázek 12. Díky použití Unit of Work v datové vrstvě bude možno při testování buissness logiky nahradit datovou vrstvu vrstvou fiktivní („in-memory“) a ověřit tak funkčnost aplikační logiky bez použití databáze při testech. Informace o tomto návrhovém vzoru naleznete na odkazech [4] a [5].

11.4 Třídní diagram

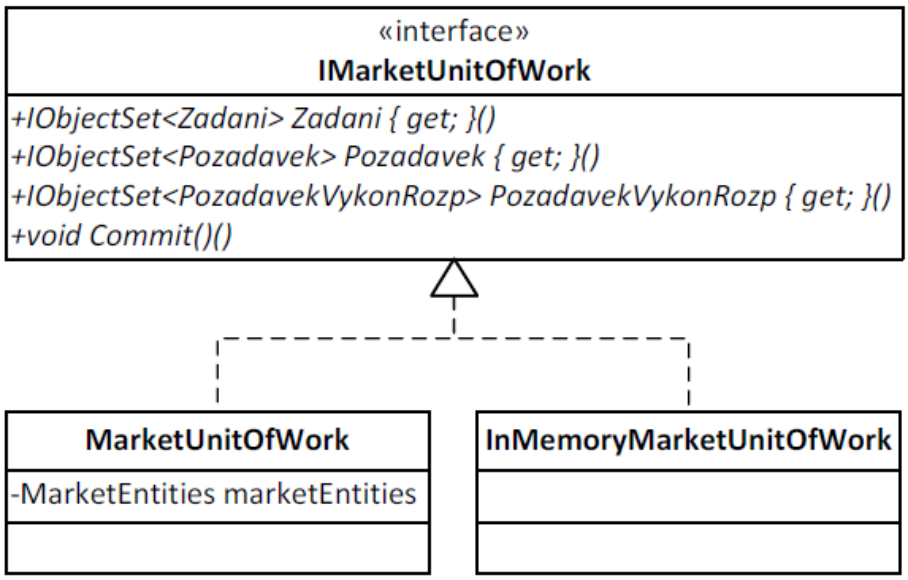
Jak je vidět z diagramu 13, srdcem celé aplikace je třída Worker, jejíž úkolem je provést samotnou synchronizaci. Je závislá na rozhraních, která plní specifické úkoly (logování, volba směru synchronizace, persistence, mapování Sprint - Zadani, synchronizace časových razítek, detekce smazaných entit...) Třídy, které tato rozhraní realizují, mohou být závislé



Obrázek 10: Třídní diagram perzistetní vrstvy pro TFS

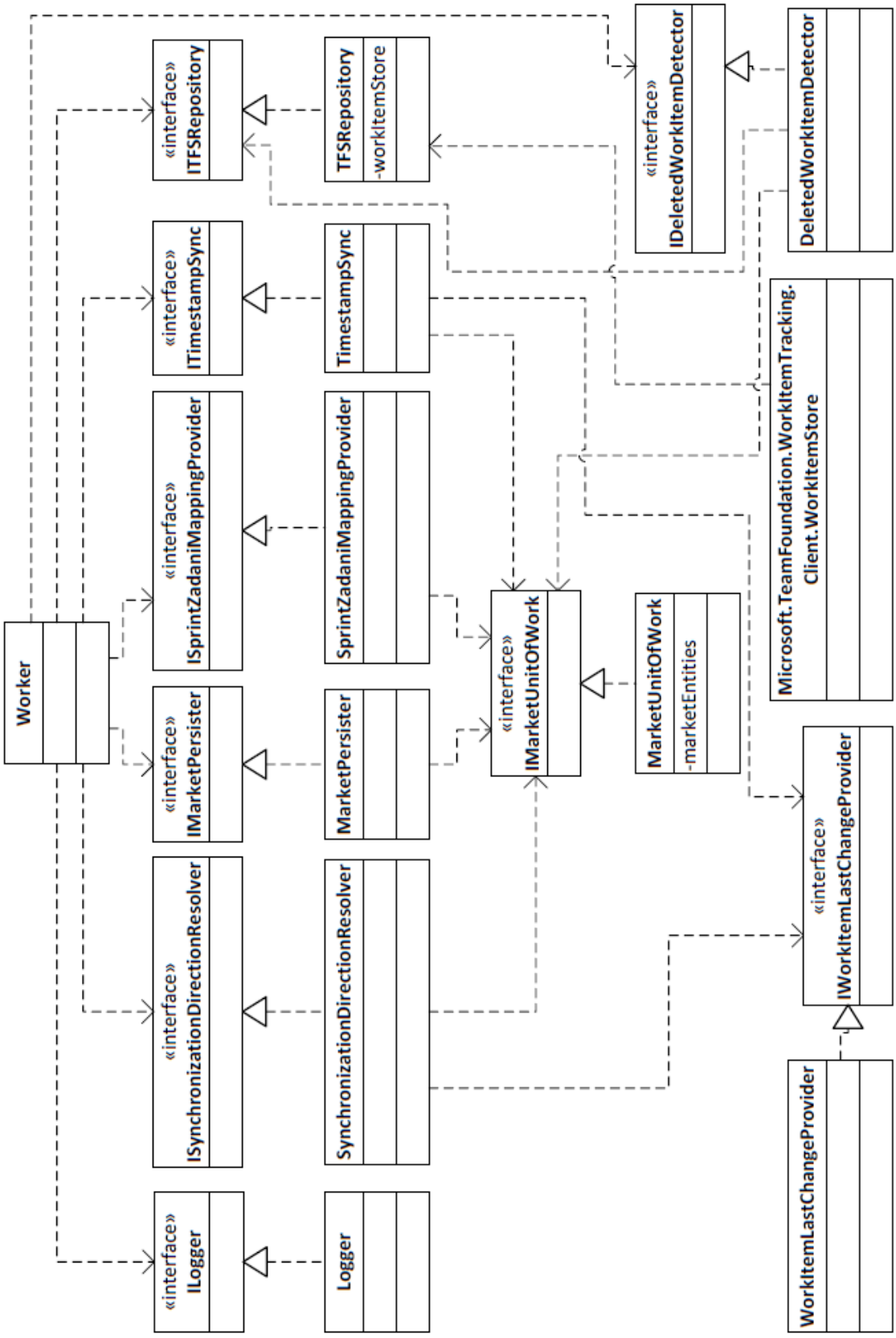


Obrázek 11: Objektový model Marketu (Entity Framework)



Obrázek 12: Návrhový vzor Unit of Work

na dalších rozhraních z knihovny. Celá aplikace tak stojí na principu Inversion of Control, konkrétně Dependency Injection, který je popsán v kapitole 11.1.



Obrázek 13: Třídní diagram aplikace

12 Implementace

Ke zhotovení nástroje byl použit programovací jazyk C# a pro demonstrační účely byl firmou doporučen konzolový projekt s tím, že veškerá funkcionalita je striktně oddělena v knihovnách. Následující odstavce popisují důležité postupy použité k implementaci a detaily klíčových bloků celého řešení.

12.1 Mapování entit Marketu na entity v TFS2010

Databáze Marketu z historických nepoužívá nastavení constraints, takže po vygenerování business modelu v Entity Frameworku je potřeba tento model doladit. Přidáme asociace, přičemž primární klíč v tabulce začíná písmeny *lID...* a pokud je tento klíč použit v jiné tabulce, začíná tam bez původního *l*, tedy *ID...* S touto vědomostí můžeme u asociací nastavit referential constraints. Výsledkem bude, že se model chová jako kdyby v databázi klíče a constraints byly používány běžným způsobem. Tento přístup nijak neovlivní jiné aplikace nad databází Marketu.

Zadávající firma počítá s tím, že budou nutné úpravy jak v databázi Market, tak ve zvolené šabloně v TFS 2010. Základem je implementovat do Team Foundation Serveru vztah 1:N mezi tabulkami Sprint-Product Backlog Item a Product Backlog Item-Task tak, jako to je v databázi Marketu (Zadani a Pozadavek, Pozadavek a PozadavekVykonRozp). To je možné provést za pomoci hierarchie mezi Work Items, která je v TFS2010 implementována. Na vrcholu hierarchie stojí Sprint, který má jako potomky jednotlivé stories (Product Backlog Items) a tyto mají za potomky úkoly (Task). Tím máme zaručenou stejnou strukturu, jaká je v Marketu. Viz tabulka 1 a obrázek 14.

Některé atributy byly kvůli lepší přehlednosti do TFS 2010 zkopírovány z Marketu a opačně. Z toho vyplývá, že před spuštěním produkční verze synchronizačního nástroje bude nutné stejným způsobem rozšířit produkční databázi Marketu a produkční TFS 2010. Následující podkapitoly popisují, jak upravit šablonu v TFS 2010 a které atributy jsou jsou aplikací brány v potaz. Market stačí upravit tak, že změníme 3 tabulky v produkční databázi podle databáze, kterou používá synchronizační nástroj jako demonstrační. Po dohodě s firmou byla zavedena jmenná konvence pro atributy používané synchronizační aplikací v Marketu - jsou to sloupce začínající písmeny *tfs*.

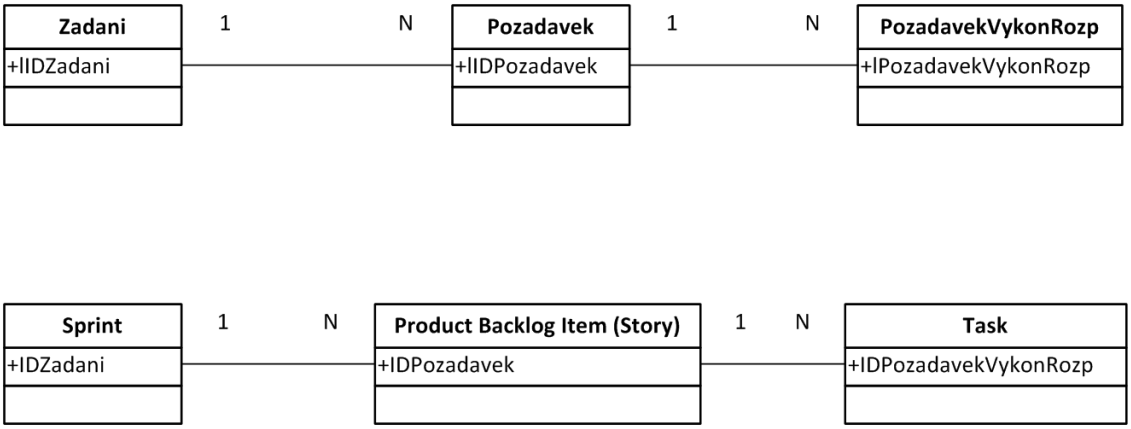
12.1.1 Úprava šablony v TFS 2010

Team Foundation Server Power Tools je sada vylepšení, nástrojů a příkazů, které zlepšují fungování TFS 2010 v rozmanitých prostředích. Pro úpravu šablon je určen nástroj zvaný Process Editor, který po instalaci balíku nalezneme ve Visual Studiu v menu Tools. Jednotlivé části šablony se dají importovat a exportovat z TFS 2010 do XML souborů a naopak. Editace je možná jak na souborech, tak přímo v TFS 2010, pokud je Visual Studio 2010 připojeno.

Pro potřeby této aplikace je potřeba upravit nebo přidat některé atributy do entit Sprint, Product Backlog Item a Task. Při přidávání atributů z Marketu nás zajímá především datový typ, který je většinou integer nebo string, a dále pole Reference name, které

Entita v Marketu	Entita v TFS2010
Zadani	Sprint
Pozadavek	Product Backlog Item
PozadavekVykonRozp	Task

Tabulka 1: Mapování entit mezi systémy



Obrázek 14: Mapování entit Marketu (první řádek) na entity v TFS 2010

později poslouží v perzistentní vrstvě jako jedinečný identifikátor přidávaného atributu (viz například výpis kódu 4).

V záložce Layout se nachází strom vizuálních prvků, kam můžeme přidávat nové prvky nebo provádět i komplexnější změny uživatelského rozhraní. Rozhodující je políčko Field Name, ve kterém musí být stejný řetězec jako v Reference Name. V poli label by měl být text, který jednoznačně popisuje daný atribut, tj. v případě sloupců z Marketu je vhodné uvést stejný název, který je v Marketu.

V poslední záložce můžeme měnit workflow dané workitem, takový požadavek ale nebyl pro tuto aplikaci předložen. Po provedení změn je nutné template ručně(!) uložit tlačítkem save, které buďto provede update v TFS 2010 nebo uloží XML soubor.

12.1.2 Sprint - Zadani

Data mezi entitami Sprint a tabulkou Zadani jsou mapována podle tabulky 2. Nové atributy v tabulce Zadani jsou zvýrazněny.

12.1.3 Product Backlog Item - Pozadavek

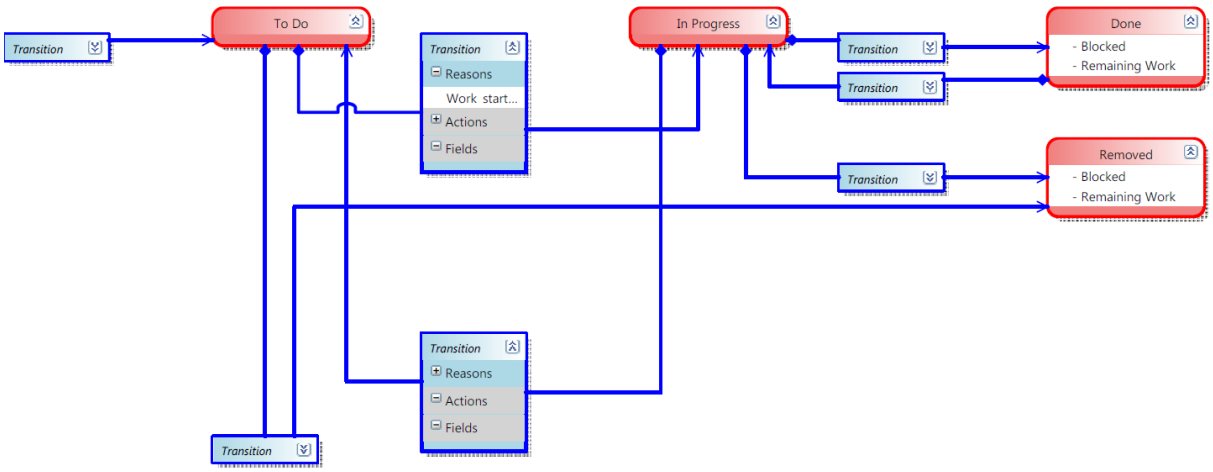
Data mezi Product Backlog Item a tabulkou Pozadavek jsou mapována podle tabulky 3. Nové atributy v tabulce Pozadavek jsou zvýrazněny.

Atribut tabulky Zadani	Datový typ	Atribut Sprintu	Poznámka
lIDZadani	int	IDZadani	identita
szPopisZadani	varchar(1000)	title	název
dtDatumAdd	datetime	StartDate	začátek sprintu
dtTermZavaz	datetime	FinishDate	konec sprintu
szUprZadani	varchar(1000)	DescriptionHTML	popis sprintu
szCislo	varchar(20)	Cislo	Sprint je rozšířen o tento atribut z Marketu
tfsWorkItemId	int	Id	id z TFS je přeneseno do Marketu
tfsSynchronize	bit	-	bit přepínač synchronizace (zapnuto/vypnuto)
tfsIterationPath	nvarchar(250)	Iteration	cesta ke sprintu
tfSTimestamp	datetime2(7)	-	časové razítko posl. změny
tfsLastSync	datetime2(7)	History - poslední změna	datum poslední změny položky

Tabulka 2: Mapování Zadani - Sprint

Atribut tabulky Pozadavek	Datový typ	Atribut PBI	Poznámka
lIDPozadavek	int	IDPozadavek	identita
szPozadavek	varchar(3000)	title	název
szPoznamka	varchar(1000)	DescriptionHTML	popis Product Backlog Item
IDPriorita	int	Priorita	PBI je rozšířen o tento atribut z Marketu
szHodina	varchar(10)	Effort	zbývající čas
tfSTimestamp	datetime2(7)	-	časové razítko posl. změny
tfsLastSync	datetime2(7)	History - poslední změna	datum poslední změny položky

Tabulka 3: Mapování Pozadavek - Product Backlog Item



Obrázek 15: Task.State workflow

12.1.4 Task - PozadavekVykonRozp

Data mezi Task a tabulkou PozadavekVykonRozp jsou mapována podle tabulky 4. Nové atributy v tabulce PozadavekVykonRozp jsou zvýrazněny.

Tady je nutné dávat pozor na atribut tfsState. Jedná se o WorkItem typu Task a jeho atribut State, který má v šabloně Scrum 1.0 předem definovaný workflow. Stavy není možné měnit libovolně, ale podle workflow diagramu číslo 15.

Změní-li uživatel v Marketu stav například z *Done* na *To Do*, při další synchronizaci dojde k chybě při ukládání příslušné entity Task, protože takováto změna není ve workflow povolena. Tuto situaci by šlo vyřešit tak, že by klientská aplikace, která přistupuje do databáze Marketu, o tomto workflow veděla a příslušný sloupec by validovala a případně upozornila uživatele, že takto není možné provést update. Jiným řešením by mohlo být ignorování tohoto sloupce triggerem i synchronizačním nástrojem tak, že by nešlo o relevantní položku a tento sloupec by se synchronizoval jen v případě směru z TFS 2010 do Marketu.

V synchronizačním nástroji je momentálně pokus o uložení takového Tasku zalogován jako chyba a pokračuje se dalším úkolem. Při dalším rozšiřování nebo údržbě synchronizačního nástroje musíme mít na paměti, že atributy tohoto typu mají v TFS 2010 vlastní workflow.

12.2 Rozšíření entit Marketu o časová razítka

Cílem implementační části této diplomové práce není vytvořit dokonalý verzovací systém, jaký má TFS 2010, ale kompaktní a funkční řešení synchronizace. Záznam v Marketu může být změněn přímo z Marketu, synchronizačním nástrojem, přímo v databázi... Kdo nebo co změnu provede není relevantní. Důležité je pouze datum poslední změny. Proto byl do tabulek Zadani, Pozadavek a PozadavekVykonRozp přidán atribut tfSTimestamp. Účelem tohoto pole je pamatovat si čas poslední změny příslušného řádku.

Atribut tabulky PozadavekVykonRozp	Datový typ	Atribut Tasku	Poznámka
lIDPozadavekVykonRozp	int	IDPozadavekVykonRozp	identita
szPoznamka	varchar(250)	title	název
fHodin	float	RemainingWork	zbývající čas
tfsState	nvarchar(20)	State	stav přenesený z Marketu
tfSTimestamp	datetime2(7)	-	časové razítko posl. změny
tfsLastSync	datetime2(7)	History - poslední změna	datum poslední změny položky

Tabulka 4: Mapování PozadavekVykonRozp - Task

```
GO
IF OBJECT_ID ('[Market].[dbo].PozadavekUpdateTrigger', 'TR') IS NOT NULL
   DROP TRIGGER PozadavekUpdateTrigger;
GO

CREATE TRIGGER PozadavekUpdateTrigger
ON [Market].[dbo].Pozadavek
AFTER INSERT, UPDATE
NOT FOR REPLICATION

-- fire only for columns in the following condition:
AS IF (UPDATE (lIDPozadavek) OR UPDATE (szCislo) OR UPDATE (szPozadavek) OR UPDATE (
     szPoznamka) OR UPDATE (szHodina) OR UPDATE (IDPriorita))
BEGIN
   -- do NOT fire if this trigger is already running or the sync tool is updating the whole row
   -- and fire when INSERT is called (lIDPozadavek is updated is this case)
   IF NOT (update(tfsTimestamp)) OR UPDATE (lIDPozadavek)
      BEGIN
         UPDATE Pozadavek
         SET tfsTimestamp = GETDATE()
         WHERE lIDPozadavek IN (SELECT lIDPozadavek FROM inserted)
      END
END
```

Výpis 6: Trigger pro tabulku Pozadavek

Jak toto pole efektivně aktualizovat? Microsoft SQL Server podporuje tzv. trigger, neboli spoušť, která je spuštěna jako reakce na určitou událost v databázi. Naším spouštěčem bude operace UPDATE, na kterou budeme reagovat akutalizací atributu tfSTimestamp. Ne všechny operace UPDATE jsou spouštěči, protože ne všechny atributy jsou relevantní v rámci synchronizace. Jsou to pouze atributy v tabulkách 2, 3 a 4 vyjma situací, kdy je aktualizován sloupec tfsTimestamp (trigger nesmí vyvolat sám sebe) nebo sloupec začínající lID... Trigger spustí i operace INSERT při vytvoření nového záznamu. Jak vypadá vytvoření triggeru například pro tabulku Pozadavek můžeme vidět ve výpisu kódu č. 6.

Při synchronizaci stačí porovnat datum poslední verze work item v TFS a hodnotu atributu tfsTimestamp k tomu abychom vědeli, kterým směrem budeme tu danou položku synchronizovat.

12.3 Volba směru synchronizace

Třída SynchronizationDirectionResolver má za úkol stanovit, kterým směrem se budou data mezi mapovanými entitami přenášet. Tato aktivita (obrázek 16) začíná defenzivně - pokud není známo datum poslední synchronizace (atribut tfsLastSync u entit z Marketu), považujeme za ně poslední časové razítko z Marketu (tfsTimestamp). V podstatě může být zvolen jeden z těchto pěti směrů synchronizace:

1. Market do TFS 2010

2. Market do TFS 2010 s konfliktem
3. TFS do Marketu
4. TFS do Marketu s konfliktem
5. žádná synchronizace

Informace budeme přenášet z Marketu do TFS 2010 v případě, kdy časové razítko entity z Marketu je větší než časové razítko entity z TFS 2010. Musíme ale detekovat případný konflikt.

Žádná synchronizace neproběhne v případě, kdy se data ani v jednom ze systémů nezměnila, tj. žádný uživatel nezměnil některý z relevantních údajů v Marketu nebo kterýkoliv atribut z entity v TFS 2010.

Směr z TFS 2010 do Marketu bude vybrán, když časové razítko entity z TFS 2010 bude větší než časové razítko entity z Marketu. I tady testujeme, jestli nedošlo ke konfliktu.

12.4 Detekce smazaných položek v TFS 2010

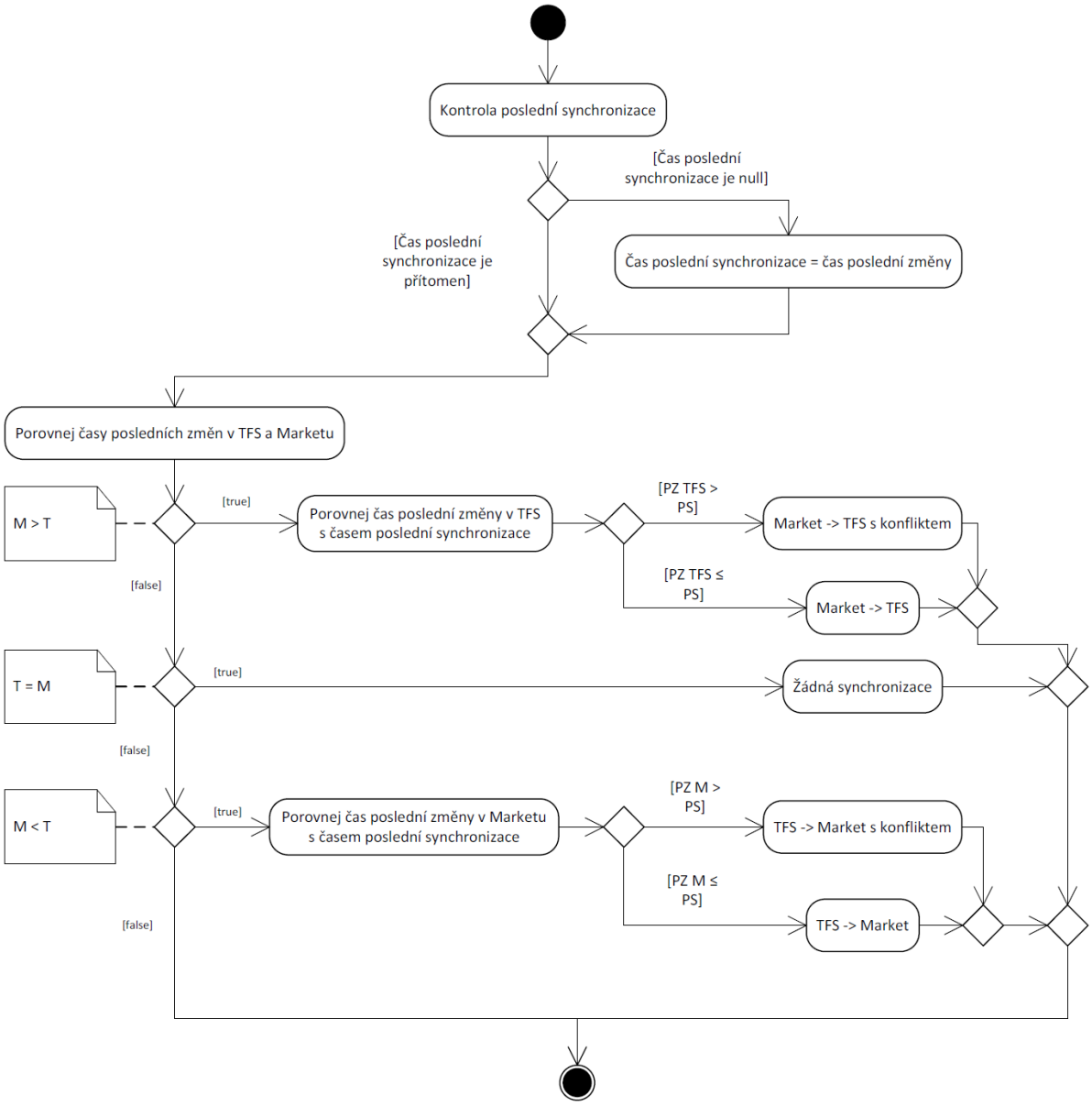
Pokud dojde ke smazání Product Backlog Item nebo Tasku v TFS 2010, musí být nástroj schopen toto identifikovat a upozornit uživatele, že nyní v Marketu existuje položka, která byla v TFS 2010 smazána. Tuto funkcionalitu představuje třída DeletedWorkItem-Detector implementující rozhraní IDeletedWorkItemDetector. Smaže-li se položka z TFS 2010 a zůstane v Marketu, jsou zbývající položky v TFS 2010 podmnožinou položek z Marketu. Hledáme tedy rozdíl, který je potom vrácen jako množina položek, které zůstaly v Marketu a byly smazány v TFS 2010.

12.4.1 Synchronizace obecně - sekvenční diagram

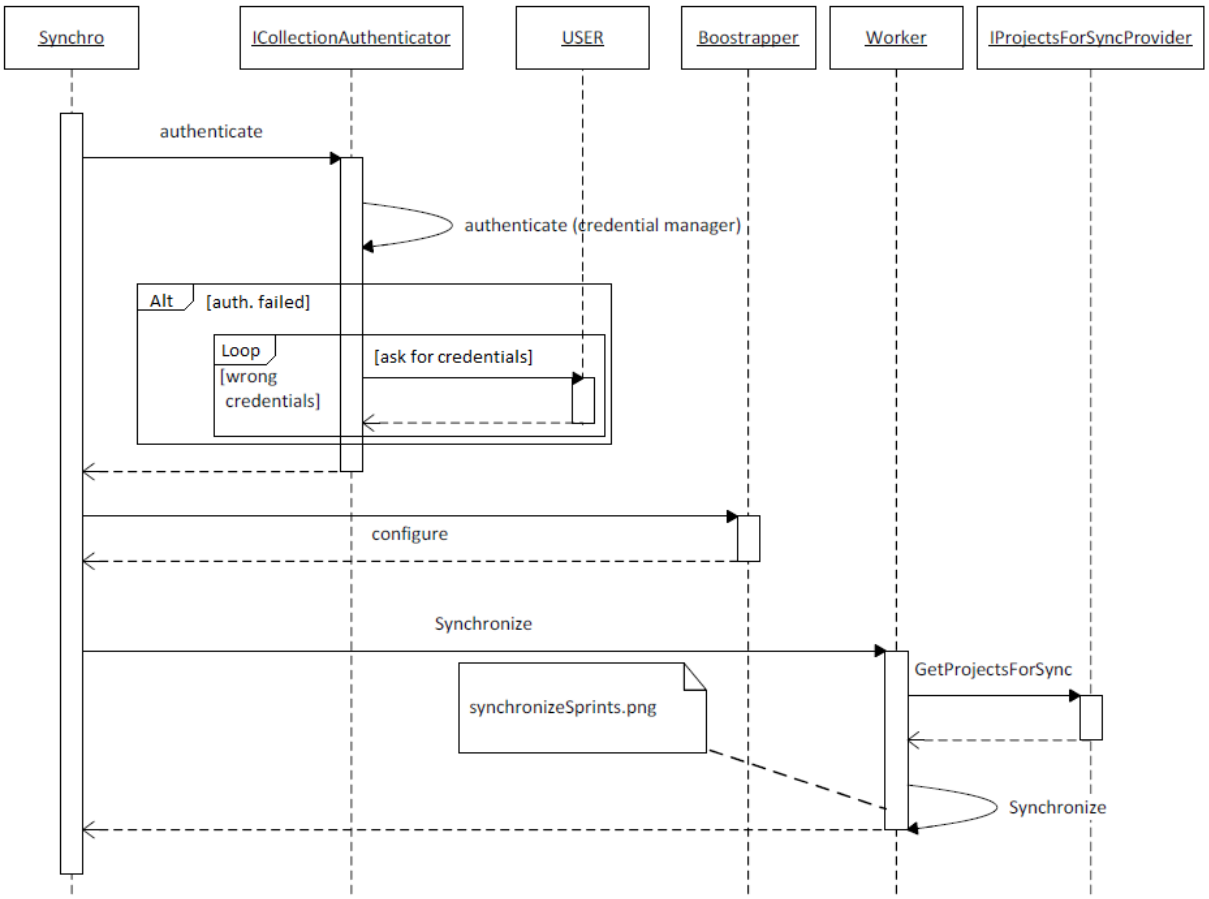
Sekvenční diagram na obrázku 17 ukazuje životní cyklus entit v aplikaci. Třída Synchro stojí mimo knihovny. Jejím úkolem je instanciovat třídu Worker, provést autentikaci a Inversion of Control pomocí třídy Boostrapper a její meotdy Configure(). Následně je spuštěna samotná synchronizace, jejíž průběh je popsán dále.

12.4.2 Synchronizace Sprintu a Zadani

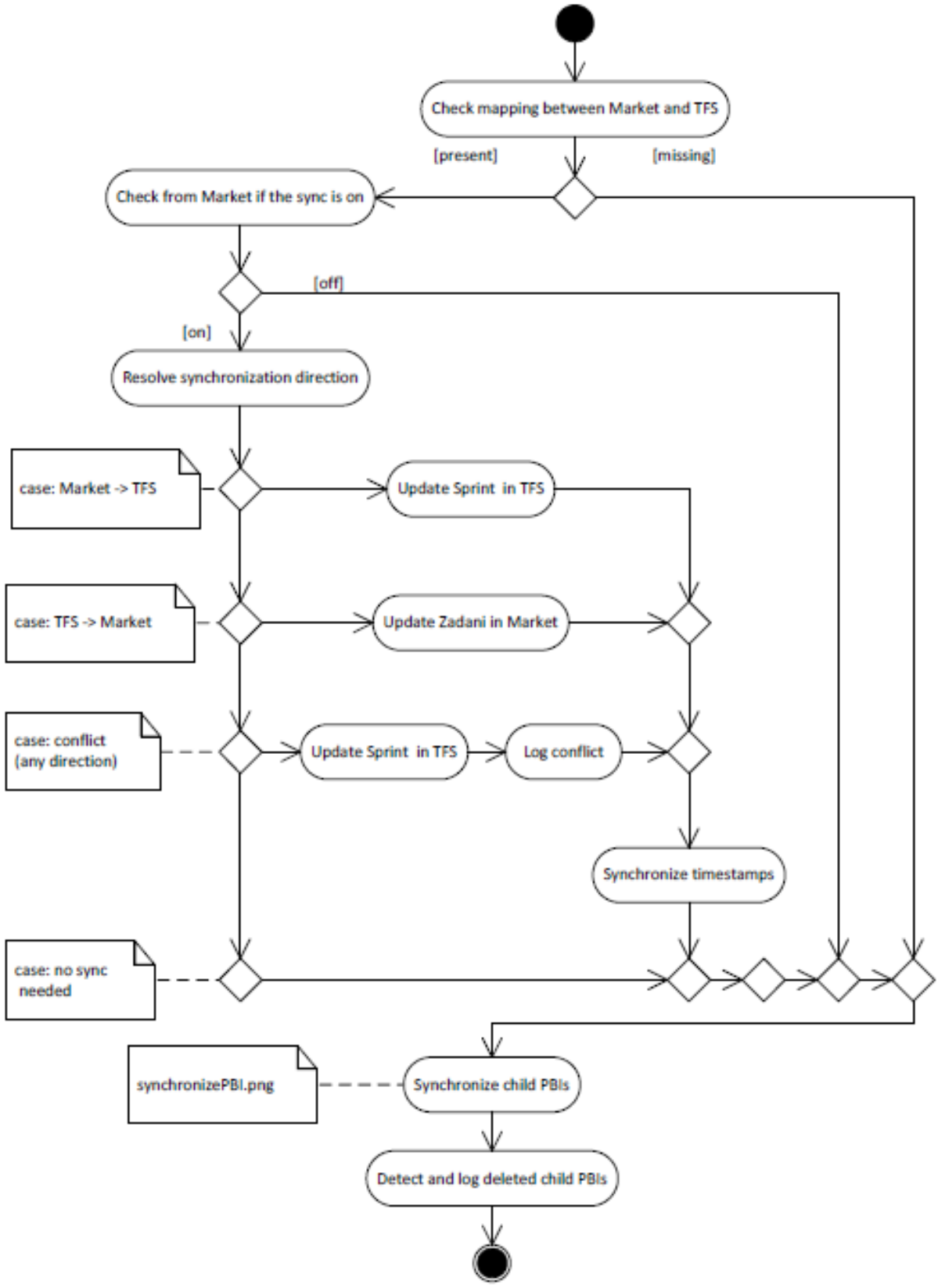
Z activity diagramu 18 lze vyčíst, že z každého projektu jsou vybrány ty sprinty, které mají správně nastavené mapování mezi TFS a Marketem a mají aktivní synchronizaci. Pokud nejsou tyto podmínky splněny, synchronizace sprintu a jakýchkoli hierarchicky podřadných entit neproběhne. Jinak se nejprve zjistí směr synchronizace, jak je popsáno v sekci 12.3. Na základě směru sychronizace jsou jednotlivé položky aktualizovány a proběhne sjednocení časových razítek, v případě konfliktu proběhne zalogování. Pokud nástroj dospěje k závěru, že synchronizace není nutná, nestane se se sprintem vůbec nic. Nicméně vždy musí být zavolána synchronizace potomků (Product Backlog Items), protože ti mohou update potřebovat.



Obrázek 16: Volba směru synchronizace



Obrázek 17: Synchronizace obecně



Obrázek 18: Synchronizace Sprintu a Zadani

12.4.3 Synchronizace entit Product Backlog Item a Pozadavek

Synchronizace Product Backlog Items a tabulky Pozadavek probíhá trochu jinak. Je to znázorněno v obrázku 19. Nejrpve se zjistí, zda PBI v TFS 2010 obsahuje IDPozadavek z Marketu. Pokud ne, jedná se o nově vytvořenou PBI a je potřeba ji vytvořit i v Marketu. Potom je nově vygenerované ID přeneseno i do TFS a provede se synchronizace časových razítek tak, aby oba časy byly stejné a při dalším běhu synchronizace už vše proběhlo běžným způsobem.

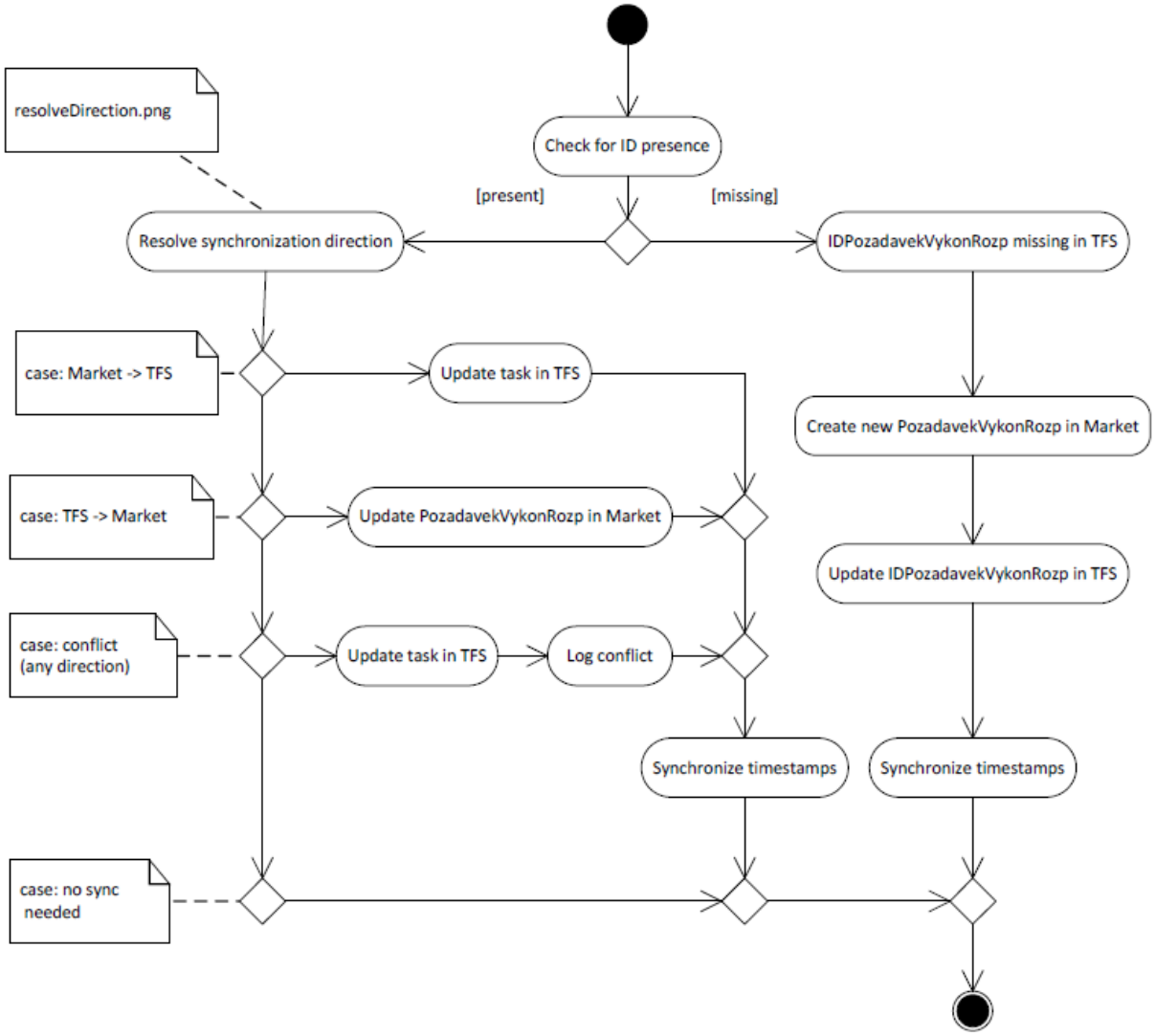
V případě, že je podle atributu IDPozadavek z TFS 2010 nalezen záznam v Marketu, může proběhnout synchronizace stejným stylem jako u Sprintu. Nakonec je spuštěna synchronizace potomků (Tasks) a to i za okolností, kdy synchronizace entit Product Backlog Item a Pozadavek neproběhla z důvodu stejných časových razítek.

12.4.4 Synchronizace entit Task a PozadavekVykonRozp

Poslední ze znázorněných průběhů synchronizace ukazuje obrázek 20. Podobně jako u Product Backlog Items, je i tady na začátku ověřeno, jestli se jedná o již existující entitu PozadavekVykonRozp. Pokud ne, je vytvořena, její ID z Marketu je přeneseno do TFS 2010 a časová razítka jsou nastavena na stejný čas.

Jinak synchronizace proběhne velice podobně jako u Sprintu nebo PBI s tím rozdílem, že před koncem aktivity už nevoláme synchronizaci na potomky, protože Task stojí na dně hierarchie a žádné nemá.

Obrázek 19: Synchronizace entit Product Backlog Item a Pozadavek



Obrázek 20: Synchronizace entit Task a PozadavekVykonRozp

13 Údržba aplikace

Za běžného provozu se předpokládá, že čas na serveru, kde má databázi TFS 2010 bude (téměř) stejný jako čas na serveru, kde má databázi Market. Je to kvůli triggerům, které při změně relevantních atributů zapisují čas změny. V případě nestejných časů by vyhodnocování směru synchronizace nefungovalo správně.

13.1 Instalace

Obrázek číslo 21 naznačuje, jak by mohlo vypadat vhodná instalace celého řešení. K TFS 2010 se připojují členové scrum týmu i scrum master přes klientské aplikace TFS, viz kapitola 3.2. Samotný synchronizační nástroj je vhodné umístit na nějaký server, klidně i tam, kde je nainstalována aplikační vrstva TFS 2010. Tyto dvě entity mezi sebou komunikují pomocí HTTP. Na jiném stroji je databázový server, který může sloužit jako datová vrstva TFS 2010 a Marketu zároveň. Někteří lidé z firmy mají možnost přistupovat do Marketu a ovlivňovat synchronizaci pomocí nastavení, která byla implementována.

13.2 Převod demo aplikace

Při samotném převodu stačí v solution vytvořit nový projekt a v něm mít stejný konfigurační soubor, jako je v konzolové aplikaci a zkopírovat třídu Boostrapper, která obsahuje konfiguraci Dependency Injection. Ve spouštěcí metodě se musí zavolat konfigurace bootstrapperu, autentikace z knihovny a zavolat metoda Worker.Synchronize, která spouští synchronizaci (viz současná spouštěcí metoda v konzolové aplikaci).

13.3 Autentikace a autorizace

13.3.1 SQL server - Market

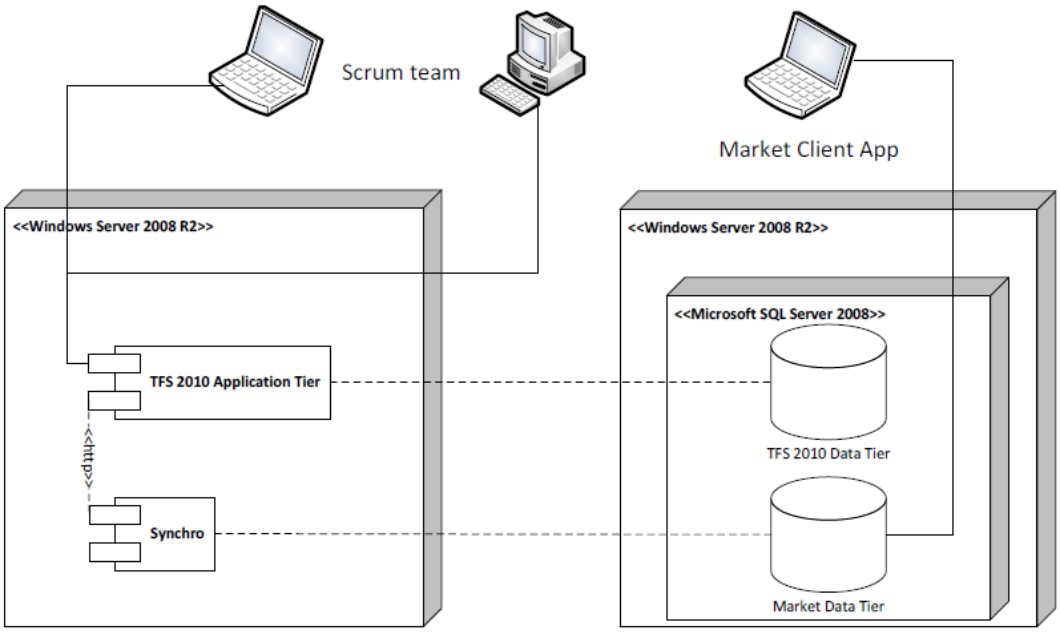
V konfiguračním souboru App.config je použit connection string s nastavením Integrated Security = true, což znamená, že se k autentikaci a autorizaci použije účet, pod kterým je synchronizační nástroj momentálně spuštěn. Musí mít potřebná oprávnění, a to i v samotné databázi Marketu (tabulky Zadani, Pozadavek, PozadavekVykonRozp).

13.3.2 TFS 2010

K autentikaci se může použít účet, pod kterým je synchronizační nástroj momentálně spuštěn. Druhou možností, pokud první varianta selže, je zadat doménu, účet a heslo do konzole.

Vybraný účet musí být autorizován pro každý projekt z TFS 2010, který chceme synchronizovat. Postup je následující:

1. Otevřete panel Team Foundation Server Administration Console
2. Otevřete příslušnou Team Project Collection



Obrázek 21: Příklad vhodné instalace celého řešení

3. Vyberte Group Membership z panelu General
4. Otevřete Project Collection Valid Users
5. Vyberte [požadovaný projekt]\Readers
6. Přidejte účet

V ideálním případě bude aplikace běžet jako Windows Service pod doménovým účtem, který bude mít příslušná oprávnění jak v SQL serveru, tak v TFS 2010.

13.4 Log

Logování v sychnronizačním nástroji probíhá nepřímo přes rozhraní ILogger a třídu Logger, která je realizuje. Tato třída používá k logování Microsoft Enterprise Library. Tento modul je snadné nakonfigurovat, pokud je nainstalovaný na počítači, ve kterém se na aplikaci pracuje. Ve Visual Studiu 2010 klikneme pravým tlačítkem na App.Config a Edit Enterprise Library V5 Configuration. V otevřeném okně vybereme Logging Settings a dokonfigurujeme logování dle potřeby. Autor práce doporučuje kromě klasického logování i zprávy přes e-mail, které by upozorňovaly nějakou zodpovědnou osobu ihned v případě konfliktu nebo chyby.

14 Technologie

Microsoft SQL Server 2008 Jak pro Market tak pro TFS 2010 byl pro demonstrační aplikaci použit SQL Server 2008. Marketu postačila Express Edition nainstalovaná na lokálním stroji, naopak TFS 2010 má na databázi daleko větší požadavky. Je potřeba nainstalovat a nakonfigurovat Analysis Services a Reporting Services s Report Server Farm, což je využito ke generování grafů. Konkrétně se jedná o verzi Enterprise Edition 10.3.5500.0.

Windows Server 2008 R2 Enterprise SP1 Protože TFS 2010 v ne-základní konfiguraci potřebuje Reporting Services, je nutné zvolit serverovou edici operačního systému Windows, která toto podporuje.

VMWare Player Pro zmíněný OS bylo vybráno virtuální prostředí VMWare Player, což umožnilo vývoj synchronizačního nástroje na stejném stroji, kde běželo IDE prostředí.

Active Directory Na virtuálním stroji byla nainstalována doména s několika účty, které jsou použity k autentikaci synchronizačního nástroje proti TFS 2010.

Team Foundation Server 2010 Tato platforma má 2 složky - datovou a aplikační. Přesto, že se z výkonnostního hlediska nedoporučuje instalovat obě složky na jeden stroj, pro vývoj synchronizačního nástroje to postačilo. Šablona MSF for Agile Software Development v5.0 je součástí instalace, Microsoft Visual Studio Scrum 1.0 je nutné stáhnout (respektive exportovat z dodané demonstrační aplikace).

Visual Studio 2010 Professional + .NET 4 Toto vývojové prostředí (IDE) bylo použito k implementaci synchronizačního nástroje, k práci s work items na testovacím projektu, k úpravě šablony Scrum 1.0, správě zdrojového kódu…Celá aplikace je napsaná v jazyce C#, .NET Framework 4.

Team Foundation Server Power Tools Průzkum, úpravy a export/import obou šablon zmíněných v této práci.

Microsoft Enterprise Library 5.0 Tato knihovna byla využita k logování.

Unity 2.0 Dependency Injection container

ADO.NET Entity Framework Persistetní vrstva Marketu

15 Licence

Jelikož sychnronizační nástroj využívá pro přístup do TFS 2010 vlastní „servisní účet“, nabízí se otázka, potřebuje-li k tomu licenci. Na to byl dotázán pan Juřek z firmy Microsoft ([8]). Cituji odpověď:

„Dobry den, licencne to nereste. Licencuji se pouze "zivi lide"a to pouze Ti, kteri vyuzivaji data TFS, ne Ti co spravuji server jako takovy. S pozdravem Michael Jurek“

Podle pravidel firmy Microsoft se takto licencují pouze „živí lidé“, proto není zvláštní licence potřeba.

16 Závěr

S firmou CID International jsem se rozhodl spolupracovat z toho důvodu, že požadovala praktické řešení problému spojeného s platformou .NET. Cílem bylo na základě analýz dvou systémů umožnit jejich paralelní chod a zefektivnit tím projektové řízení. Závěrem bych se chtěl podělit o zkušenosti nabrané během tvorby této diplomové práce, a to hned z několika úhlů pohledu.

16.1 Práce s TFS 2010

Protože Scrum je masivně používaná metodika, bylo snadné nalézt potřebné informace, nehledě na to, že jsem sám byl členem dvou Scrum týmů. Jelikož Team Foundation Server 2010 je robustní platforma, o které jsem v době zadání práce moc netušil, bylo pro mě toto téma zpočátku těžko uchopitelné. Až reálná práce se šablonami pro podporu projektového řízení mi umožnila šablony popsat. Složitější bylo porovnání obou šablon, protože jsem osobně projekt nikdy nevedl. Nicméně, pokusil jsem se podívat se na celou věc z pohledů Scrum mastera, který je v praxi za projekty zodpovědný a obvykle je velmi zaneprázdněn, a z pohledu člena Scrum týmu, kterého vyplňování formulářů pro management obvykle obtěžuje. Abych v této fázi oddělil subjektivní od objektivního, srovnal jsem nejpodstatnější činnosti, které se během spritnů provádí, a na základě toho nakonec vybral vhodnější šablonu.

16.2 Práce s Marketem

Abych mohl implementovat synchronizační nástroj, bylo mi firmou poskytnuto torzo databáze IIS Market, tj. pouze perzistentní vrstva. Jedná se o středně velkou databázi, ve které nebylo snadné se zorientovat i přes dodanou dokumentaci. K dokončení objektové analýzy bylo potřeba několik konzultací s firmou.

16.3 Průběh implementace

Nečekaným zádrhelem byla instalace všech serverů a vývojového prostředí obecně, která zabrala několik dní času. Střídavě jsem nespočetněkrát instaloval a deinstaloval jak SQL Server s Reporting Services, tak TFS 2010. Problémem byl chybějící soubor v SQL Serveru, který byl požadován až při instalaci TFS. Nakonec jsem se na fórech Microsoftu stal úspěšným řešitelem tohoto problému pro více bezradných developerů. Čas strávený řešením těchto problémů se bohužel neprojevil ani v textu práce, ani v odevzdaném synchronizačním nástroji.

Před samotnou implementací mi bylo doporučeno zpracovat obecné možnosti synchronizace a pokusit se aplikovat jejich principy ve své práci. To se později příliš neprojevilo, protože zadávající firma vyslovila takové požadavky, které v podstatě výslednému řešení určovaly směr (možnost definovat konkrétní sprinty k synchronizaci, řešení konfliktu, ...) Některé požadavky vyvstaly až během programování, takže se v podstatě jednalo o iterativní vývoj, kde byla postupně funkcionalita aplikace s časem rozšiřována.

Od firmy jsem dostal poměrně volnou ruku co se jejich IIS Market týkalo - měl jsem povoleno přidat nové atributy do sloupců, což mě nakonec dovedlo k řešení konfliktů pomocí časových razítek a posledního času synchronizace. Jak se však později ukázalo, nebyl jsem první na světě, kdo tento mechanismus navrhnul a implementoval.

Během implementace jsem využil znalostí prostředí .NET a několika softwarových návrhových vzorů, přiučil se několika novým věcem, jako jazyku WIQL, práci s knihovnami Microsoft.TeamFoundation, editaci šablon, … Vznikla dobře rozšiřitelná a udržovatelná aplikace, jak to ostatně zadávající firma chtěla. Očekává se, že text této práce poslouží především potenciálnímu developerovi, který synchronizační nástroj převezme. V neposlední řadě bylo nutné sepsat technologické požadavky, prezentovat systémový pohled na celou věc a vyřešit otázku licencování nového účtu v TFS 2010, který bude pro synchronizaci používán.

16.4 Budoucnost práce

Jedním z vylepšení, která jsem zadávající firmě navrhoval, bylo mapování uživatelů z TFS 2010 na uživatele z Marketu, aby i toto mohlo být synchronizováno. Market však neobsahuje informaci o tom, jaký má ten který uživatel doménový účet, s čímž TFS 2010 pracuje. Další směr, kam by se aplikace mohla rozšiřovat, je synchronizace více atributů, nebo přidávání více atributů z jednoho systému do druhého. Na základě logovaných konfliktů by také bylo možné vybudovat uživatelské rozhraní s nabídkou jejich řešení. Co se šablon týká, mohly by v případě potřeby být více přizpůsobeny potřebám firmy. Celé řešení je velmi pružné.

16.5 Evaluace

Tato práce analyzuje jak IIS Market, tak TFS 2010 včetně jeho šablon pro podporu projektového řízení, a byla tak splněna teoretická část. Na základě toho jsem implementoval demonstrační verzi synchronizačního nástroje, která je plně funkční a po důsledném odladění bude možné ji nasadit do ostrého provozu. Poté budou moci Scrum týmy začít využívat TFS 2010 pro projektové řízení bez dopadů na funkcionalitu Marketu a efektivita projektového řízení bude skutečně zvýšena.

17 Reference

[1] Gustafsson, Thomas; Hammarberg, Erik
A Mobile Unit Synchronization Algorithm [online]. Göteborg, Sweden, 2011 [cit. 2012-04-03]. Report. Chalmers University of Technology. Dostupné z:
`http://publications.lib.chalmers.se/cpl/record/index.xsql?pubid=143634`

[2] Saito, Yasushi; Shapiro, Marc
Optimistic replication [online, báze dat EBSCOhost]. New York, NY, USA: ACM New York, 2005 [cit. 2012-04-03]. ACM Computing Surveys (CSUR), Volume 37 Issue 1,. Dostupné z:
`http://dl.acm.org/citation.cfm?id=1057980`

[3] Bjork, Aaron
Announcing Microsoft Visual Studio Scrum 1.0. [online]. 2010[cit. 2012-04-03]. Dostupné z: http://blogs.msdn.com/b/aaronbjork/archive/2010/07/19/ announcing-microsoft-visual-studio-scrum-1-0.aspx

[4] Allen, Scott
Testability and Entity Framework 4.0. [online]. 2010[cit. 2012-04-03]. Dostupné z:
`http://msdn.microsoft.com/en-us/library/ff714955.aspx`

[5] Fowler, Martin
Unit of Work. [online]. 2012[cit. 2012-04-03]. Dostupné z:
`http://martinfowler.com/eaaCatalog/unitOfWork.html`

[6] Fowler, Martin
Inversion of Control Containers and the Dependency Injection pattern. [online]. 2004[cit. 2012-04-03]. Dostupné z:
`http://martinfowler.com/articles/injection.html`

[7] Hieatt, Edward; Mee, Rob
Repository. [online]. 2012[cit. 2012-04-03]. Dostupné z:
`http://martinfowler.com/eaaCatalog/repository.html`

[8] Jurek, Michael
E-mailová komunikace. [online]. 2012[cit. 2012-01-05]. Email:
`Michael.Jurek@microsoft.com`

[9] Skupina autorů
Manifesto for Agile Software Development. [online]. 2001[cit. 2011-10-10]. Dostupné z:
`http://agilemanifesto.org`

[10] MSDN
Extending Team Foundation. [online]. [cit. 2012-04-03]. Dostupné z:
`http://msdn.microsoft.com/library/bb130146.aspx`

[11] MSDN
MSF for Agile Software Development v5.0 [online]. [cit. 2012-03-28]. Dostupné z:
`http://msdn.microsoft.com/en-us/library/dd380647.aspx`

[12] MSDN
Agile Principles and Values [online]. [cit. 2012-03-28]. Dostupné z:
`http://msdn.microsoft.com/en-us/library/dd997578.aspx`

[13] Scrum Alliance
Scrum:The Basics, Scrum 101 [online]. [cit. 2012-03-28]. Dostupné z:
`http://www.scrumalliance.org`